

2

NAVAL POSTGRADUATE SCHOOL

Monterey, California



DTIC COPY

DTIC
ELECTE
MAR 5 1991
S B D

THESIS

AD-A232 014

A METHOD FOR MACHINERY CONDITION MONITORING
OF TRANSIENT PHENOMENA USING THE
PSEUDO WIGNER-VILLE DISTRIBUTION

by

Graham W. Rossano

June 1990

Thesis Advisor

Y.S. Shin

Co-Advisor

J.F. Hamilton

Approved for public release; distribution is unlimited

91 2 28 054

Unclassified

security classification of this page

REPORT DOCUMENTATION PAGE				
1a Report Security Classification Unclassified			1b Restrictive Markings	
2a Security Classification Authority			3 Distribution Availability of Report	
2b Declassification Downgrading Schedule			Approved for public release; distribution is unlimited.	
4 Performing Organization Report Number(s)			5 Monitoring Organization Report Number(s)	
6a Name of Performing Organization Naval Postgraduate School		6b Office Symbol (if applicable) 34	7a Name of Monitoring Organization Naval Postgraduate School	
6c Address (city, state, and ZIP code) Monterey, CA 93943-5000			7b Address (city, state, and ZIP code) Monterey, CA 93943-5000	
8a Name of Funding Sponsoring Organization		8b Office Symbol (if applicable)	9 Procurement Instrument Identification Number	
8c Address (city, state, and ZIP code)			10 Source of Funding Numbers	
			Program Element No	Project No Task No Work Unit Accession No
11 Title (include security classification) A METHOD FOR MACHINERY CONDITION MONITORING OF TRANSIENT PHENOMENA USING THE PSEUDO WIGNER-VILLE DISTRIBUTION				
12 Personal Author(s) Graham W. Rossano				
13a Type of Report Master's Thesis		13b Time Covered From To	14 Date of Report (year, month, day) June 1990	15 Page Count 172
16 Supplementary Notation The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
17 Cosati Codes			18 Subject Terms (continue on reverse if necessary and identify by block number)	
Field	Group	Subgroup	machinery monitoring, transient, pseudo wigner-ville distrib.	
19 Abstract (continue on reverse if necessary and identify by block number) The Pseudo Wigner-Ville Distribution is a time-frequency representation of an input time signal and is ideally suited for portraying non-stationary signals. A working computer program is presented and the effect of preprocessing and postprocessing data manipulations is shown. The program has been developed for analyzing data for use in machinery condition monitoring and diagnostics and will be a valuable asset for analyzing transient machinery. A practical example showing pump speed variations with time is also presented. Due to the fact that the Pseudo Wigner-Ville Distribution can be used to analyze both steady state and transient operations, along with the fact that it can be calculated on virtually any computer, this method could revolutionize machinery condition monitoring and diagnostics.				
20 Distribution Availability of Abstract <input checked="" type="checkbox"/> unclassified unlimited <input type="checkbox"/> same as report <input type="checkbox"/> DTIC users			21 Abstract Security Classification Unclassified	
22a Name of Responsible Individual Y.S. Shin			22b Telephone (include Area code) (408) 646-2568	22c Office Symbol 69Sg

DD FORM 1473, 84 MAR

83 APR edition may be used until exhausted
All other editions are obsolete

security classification of this page

Unclassified

Approved for public release; distribution is unlimited.

A Method for Machinery Condition Monitoring
of Transient Phenomena using the
Pseudo Wigner-Ville Distribution

by

Graham W. Rossano
Lieutenant, United States Navy
B.S.M.E., U.S. Naval Academy, 1983

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

from the

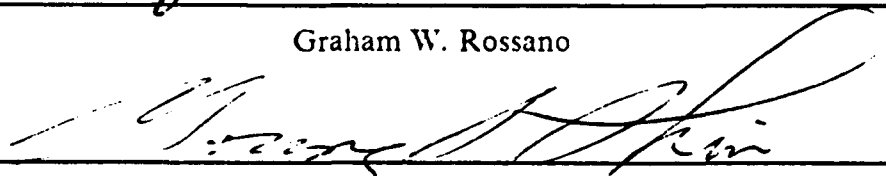
NAVAL POSTGRADUATE SCHOOL
June 1990

Author:

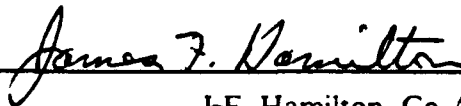


Graham W. Rossano

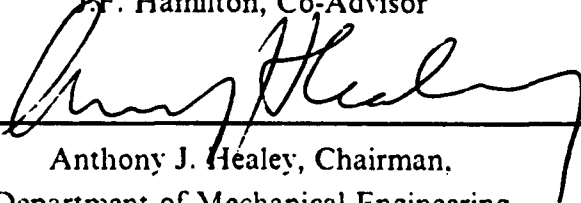
Approved by:



Y.S. Shin, Thesis Advisor



J.F. Hamilton, Co-Advisor



Anthony J. Healey, Chairman,
Department of Mechanical Engineering

ABSTRACT

The Pseudo Wigner-Ville Distribution is a time-frequency representation of an input time signal and is ideally suited for portraying non-stationary signals. A working computer program is presented and the effect of preprocessing and postprocessing data manipulations is shown. The program has been developed for analyzing data for use in machinery condition monitoring and diagnostics and will be a valuable asset for analyzing transient machinery. A practical example showing pump speed variations with time is also presented. Due to the fact that the Pseudo Wigner-Ville Distribution can be used to analyze both steady state and transient operations, along with the fact that it can be calculated on virtually any computer, this method could revolutionize machinery condition monitoring and diagnostics.

Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By _____	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	



THESIS DISCLAIMER

The reader is cautioned that computer programs developed in this research may not have been exercised for all cases of interest. While every effort has been made, within the time available, to ensure that the programs are free of computational and logic errors, they cannot be considered validated. Any application of these programs without additional verification is at the risk of the user.

TABLE OF CONTENTS

I. INTRODUCTION	1
II. THE PSEUDO WIGNER-VILLE DISTRIBUTION	4
A. WIGNER DISTRIBUTION FUNCTION	4
B. WIGNER-VILLE DISTRIBUTION	6
C. PSEUDO WIGNER-VILLE DISTRIBUTION	7
III. THE WIGFUN COMPUTER PROGRAMS	9
A. BRIEF DESCRIPTION	9
B. LIMITING DECISIONS	17
C. WIGFUN1	18
1. Effect of a Mean Value	18
2. Effect of Amplification of Time Signal Amplitude	22
3. Effect of Time Signal Windows	26
4. Effect of Making a Real Signal Analytic	34
D. WIGFUN1B	37
E. WIGFUN2	37
1. Effect of Changing the Definition of the Wigner Distribution	37
F. WIGFUN3	37
1. Effect of Reducing the Wigner-Ville Distribution	38
2. Effect of Smoothing the Wigner-Ville Distribution	42
G. WIGFUN4A, WIGFUN4B, AND WIGFUN4C	47
H. OTHER CONSIDERATIONS	47
1. Effect of Noise	47
2. Distribution Interpretation	54
3. Swept Sine Wave Example	59
IV. PRACTICAL EXAMPLES USING PSEUDO WIGNER-VILLE DISTRIBUTION	66
V. CONCLUSIONS	85

VI. RECOMMENDATIONS FOR FUTURE RESEARCH	86
APPENDIX A. COMPUTER CODE	87
A. SUMMARY	87
B. FORTRAN PROGRAMS	91
C. ALPHANUMERIC LISTING OF SUBROUTINES	109
D. DATA CONVERSION PROGRAM	150
APPENDIX B. DATA TRANSFER FROM SOURCE TO VAX COMPUTER	152
A. DATA SOURCE TO HP3565 COMPUTER	152
B. HP3565 COMPUTER TO PC	152
C. PC TO VAX	152
D. EDITING DATA FILE UPON ARRIVAL IN THE VAX	152
LIST OF REFERENCES	154
BIBLIOGRAPHY	157
INITIAL DISTRIBUTION LIST	160

LIST OF FIGURES

Figure 1. Time and Frequency Domains	2
Figure 2. Flow Chart of WIGFUN Computer Programs	9
Figure 3. Pseudo Wigner-Ville Distribution of Two Sine Waves	11
Figure 4. Time Signal for Two Sine Waves	12
Figure 5. Pseudo Wigner-Ville Distribution of a Linear Chirp	13
Figure 6. Time Signal for Linear Chirp	14
Figure 7. Different Viewing Perspectives	15
Figure 8. Detailed Contour Plot of Linear Chirp	16
Figure 9. Linear Chirp with Mean Value in the Time Domain	19
Figure 10. Linear Chirp with Mean Value Removed from the Time Domain	20
Figure 11. Time Signal for Linear Chirp	21
Figure 12. No Time Signal Amplitude Amplification	23
Figure 13. Time Signal Amplitude Amplification of 4.0	24
Figure 14. Time Signal for 400 Hz Sine Wave in Noise	25
Figure 15. Pseudo Wigner-Ville Distribution with No Window	28
Figure 16. Input Time Signal	29
Figure 17. Pseudo Wigner-Ville Distribution with Hanning Window	30
Figure 18. Input Time Signal Modified by Hanning Window	31
Figure 19. Pseudo Wigner-Ville Distribution with Modified Hamming Window ...	32
Figure 20. Input Time Signal Modified by Modified Hamming Window	33
Figure 21. Pseudo Wigner Distribution	35
Figure 22. Analytic Time Signal	36
Figure 23. Pseudo Wigner-Ville Distribution Reduced to 64 x 32 Points	39
Figure 24. Pseudo Wigner-Ville Distribution Reduced to 128 x 64 Points	40
Figure 25. Pseudo Wigner-Ville Distribution Reduced to 256 x 128 Points	41
Figure 26. Linear Chirp with No Smoothing	43
Figure 27. Linear Chirp with Smoothing	44
Figure 28. Different Views of Linear Chirp with No Smoothing	45
Figure 29. Detailed Contours of Linear Chirp with No Smoothing	46
Figure 30. Pseudo Wigner-Ville Distribution with Minimal Noise	48
Figure 31. Input Time Signal with Minimal Noise	49

Figure 32. Pseudo Wigner-Ville Distribution with Noise	50
Figure 33. Input Time Signal with Noise	51
Figure 34. Pseudo Wigner-Ville Distribution with More Noise	52
Figure 35. Input Time Signal with More Noise	53
Figure 36. Pseudo Wigner-Ville Distribution	55
Figure 37. Input Time Signal	56
Figure 38. Pseudo Wigner-Ville Distribution	57
Figure 39. Input Time Signal	58
Figure 40. Swept Sine Wave, Frequency Span 0-1024 Hz	60
Figure 41. Detailed Contour Plot Swept Sine Wave, Frequency Span 0-1024 Hz ...	61
Figure 42. Input Time Signal for Swept Sine Wave, Frequency Span 0-1024 Hz ...	62
Figure 43. Swept Sine Wave, Frequency Span 0-512 Hz	63
Figure 44. Detailed Contour Plot Swept Sine Wave, Frequency Span 0-512 Hz ...	64
Figure 45. Input Time Signal for Swept Sine Wave, Frequency Span 0-512 Hz ...	65
Figure 46. Pump Speed Steady State 10 GPM	67
Figure 47. Detailed Contours of Steady State 10 GPM	68
Figure 48. Input Time Signal Steady State 10 GPM	69
Figure 49. Pump Speed Steady State 100 GPM	70
Figure 50. Detailed Contours of Steady State 100 GPM	71
Figure 51. Input Time Signal Steady State 100 GPM	72
Figure 52. Pump Transient Speed Up	73
Figure 53. Detailed Contours of Speed Up	74
Figure 54. Multiple Views of Speed Up	75
Figure 55. Input Time Signal of Speed Up	76
Figure 56. Pump Transient Coast Down	77
Figure 57. Detailed Contours of Coast Down	78
Figure 58. Multiple Views of Coast Down	79
Figure 59. Input Time Signal of Coast Down	80
Figure 60. Pump Transient Coast Down and Speed Up	81
Figure 61. Detailed Contours of Coast Down and Speed Up	82
Figure 62. Multiple Views of Coast Down and Speed Up	83
Figure 63. Input Time Signal of Coast Down and Speed Up	84

ACKNOWLEDGMENT

I would like to express my appreciation to all those who made this work possible. In particular, my thesis advisor, Professor Shin and my co-advisor, Professor Hamilton were always able to provide guidance and keep things in perspective. Alan Pride, Ruth Holtzman, Debbie Cuomo, and Bill McInnis all helped in identifying a problem which needed solving. Finally, to Dr. Kam Ng who pointed me to the Wigner Distribution.

I. INTRODUCTION

The physical condition or state of health of machineries which operate in short duration cycles is not known with any degree of accuracy. Maintenance on these machineries is being conducted periodically in order to avoid failures and prolong the useful operating life of the equipment. These machineries, since they operate for only short periods of time, can be characterized as transient. Additionally, machineries which are not operating in a steady state condition are also transient. This includes rotating machinery transitioning from one speed to another.

In order to assess the physical condition of machinery without complete disassembly, a physical measurement of its vibrations is conducted using an accelerometer. Other sensors, such as temperature or pressure transducers, could also be used. There are other methods, including motor current signature analysis on electrically driven machinery and wear debris analysis which could be used.[Ref. 1] However, vibrations are used predominantly for machinery condition monitoring. The vibrations are recorded in the time domain.

The time domain representations of vibrations may be decomposed into a summation of sine waves and can be identified in the frequency domain (see Figure 1 on page 2 [Ref. 2]). As long as the physical characteristics of the machinery are known, these sine waves or frequency components can be directly attributed to physical events occurring within the machinery. For example, a shaft rotating at 3600 RPM (60 Hz) with a 10 tooth spur gear will have a gear mesh frequency of 600 Hz ($10 \times 60 = 600$). Therefore a frequency component of 600 Hz may be attributed to the gear. As the speed of the shaft changes the gear mesh frequency will also change. All physical components, including bearings, couplings, impellers, rotors, etc., may be related in a similar manner to frequencies dependent upon the shaft rotation speed. Therefore, for transient machinery, as speeds vary with time the frequency components will also vary.

There is a need for a method to represent the time dependent events which occur with machinery operating in transient modes. At each instant in time as the speed of the machinery changes the frequency content will also change. The Pseudo Wigner-Ville Distribution is the method which was chosen to portray these time dependent changes.

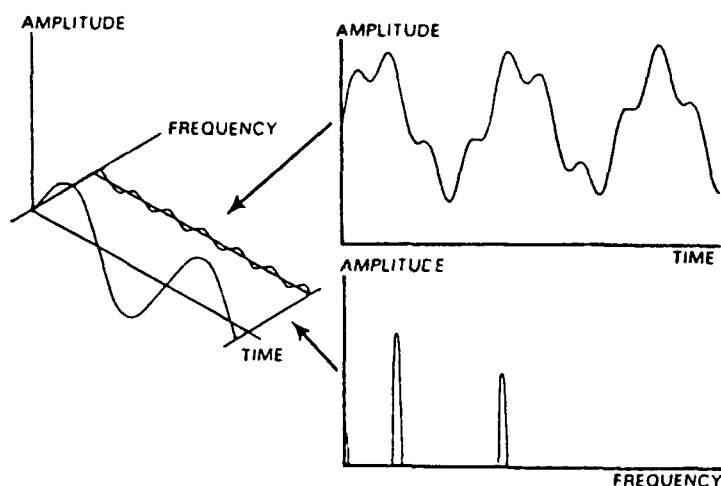


Figure 1. Time and Frequency Domains

The Pseudo Wigner-Ville Distribution is a three dimensional (time, frequency, and amplitude) representation of an input signal and is ideally suited for portraying transient phenomena. The Wigner Distribution has been used in the areas of optics [Refs. 3, 4, 5] and speech [Refs. 6, 7]. Wahl and Bolton [Ref. 8] are using it to identify structure-borne noise components. Flandrin et. al. [Ref. 9] recently proposed its use in the area of machinery condition monitoring and diagnostics, while Forrester [Ref. 10] is investigating its use in gear fault detection.

The Pseudo Wigner-Ville Distribution can be used to portray both transient non-stationary phenomena as well as stationary phenomena and therefore can be used for machinery condition monitoring of all machinery. This includes machinery operating in a steady state condition. Due to the time independent nature of steady state operating conditions, the frequency content will be constant for all time. The benefits of using the Pseudo Wigner-Ville Distribution and monitoring all machinery are enormous. Machinery which has never before been monitored now can be. Additionally, monitoring now is not limited to just a given steady state operating condition, all speeds can be monitored, the effects of different modes of vibration can be investigated, and a more thorough evaluation of the machinery condition can be obtained. This all translates into a tremendous economic savings. Here-to-fore unmonitored machinery now can be monitored and machinery monitoring is not limited to just steady state operating conditions.

The WIGFUN computer programs presented require that data be collected in the time domain and be digitized. Once it is digitized, then virtually any computer can take the digitized data and analyze it. The only hardware required is a transducer with a power supply, an analog to digital converter, and a computer.

II. THE PSEUDO WIGNER-VILLE DISTRIBUTION

A. WIGNER DISTRIBUTION FUNCTION

The Wigner Distribution Function (WDF) was first introduced by E. Wigner in 1932 [Ref. 11]. Claassen and Mecklenbrauker, in a three part series of papers, developed mathematical equations for the WDF in continuous time [Ref. 12] and discrete time [Ref. 13]. They also showed relations with other time-frequency transformations [Ref. 14]. For the continuous time case using two different complex signals, $r(t)$ and $s(t)$, the cross-Wigner Distribution can be formed. The cross-Wigner Distribution is defined as:

$$WDF_{r,s}(t, \omega) = \int_{-\infty}^{\infty} e^{-j\omega\tau} r(t + \frac{\tau}{2}) s^*(t - \frac{\tau}{2}) d\tau \quad (1)$$

where:

$r = r(t)$; a complex time signal
 $s = s(t)$; a complex time signal
 t = time
 ω = frequency
 $*$ = complex conjugate

The auto-Wigner distribution is defined as:

$$WDF_{s,s}(t, \omega) = \int_{-\infty}^{\infty} e^{-j\omega\tau} s(t + \frac{\tau}{2}) s^*(t - \frac{\tau}{2}) d\tau \quad (2)$$

Since the objective of this research is to accurately portray the time dependent frequency characteristics of a single input signal, only the auto-Wigner distribution will be used. From the frequency domain, the auto-Wigner distribution is defined as:

$$WDF_{s,s}(\omega, t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{j\xi t} S(\omega + \frac{\xi}{2}) S^*(\omega - \frac{\xi}{2}) d\xi \quad (3)$$

where:

$S = S(\omega)$; Fourier Transform of $s(t)$

There are several properties of the WDF which are important to note. As may be seen above, the WDF in the frequency domain and the WDF in the time domain are related as follows:

$$WDF_{s,s}(\omega, t) = WDF_{s,s}(t, \omega) \quad (4)$$

A time shift of a signal is a time shift of the WDF:

$$WDF_{s(t-\tau), s(t-\tau)}(t, \omega) = WDF_{s,s}(t - \tau, \omega) \quad (5)$$

A frequency shift of a signal is a frequency shift of the WDF:

$$WDF_{e^{j\Omega t}s}, e^{j\Omega t}s(t, \omega) = WDF_{s,s}(t, \omega - \Omega) \quad (6)$$

It follows that a time and frequency shift of a signal is both a time and frequency shift of the WDF:

$$WDF_{e^{j\Omega t}s(t-\tau), e^{j\Omega t}s(t-\tau)}(t, \omega) = WDF_{s,s}(t - \tau, \omega - \Omega) \quad (7)$$

Equation (7) is extremely important when we consider that these changes in time and frequency are exactly what we want to portray for our use in characterizing transient phenomena for machinery condition monitoring. It has been shown that the WDF can discriminate the frequency content of a signal at discrete times.

Integrating the WDF over time, frequency, and both time and frequency provides signal energy information. The integral of the WDF over frequency at a specific time yields the instantaneous signal power at that time:

$$\frac{1}{2\pi} \int_{-\infty}^{\infty} WDF_{s,s}(t, \omega) d\omega = |s(t)|^2 \quad (8)$$

The integral of the WDF over time at a specific frequency yields the energy density spectrum of a signal at that frequency:

$$\int_{-\infty}^{\infty} WDF_{s,s}(t, \omega) dt = |S(\omega)|^2 \quad (9)$$

The integral of the WDF over the whole plane, both time and frequency, yields the total energy in the signal:

$$\frac{1}{2\pi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} WDF_{s,s}(t, \omega) dt d\omega = ||s||^2 \quad (10)$$

The computer program listed in Appendix A does not accurately maintain the energy information in the original input signal. In the preprocessing program the input time signal is windowed and amplified. In the postprocessing program the distribution is averaged over both time and frequency. Additional properties of the WDF are listed in references 12 and 15.

The discrete time auto-Wigner distribution as developed by Claasen and Mecklenbrauker [Ref. 13] is defined as:

$$WDF_{s,s}(t, \omega) = 2 \sum_{\tau=-\infty}^{\infty} e^{-j2\omega\tau} s(t+\tau) s^*(t-\tau) \quad (11)$$

The time and frequency shift properties of the continuous time WDF described by equations (5), (6), and (7) remain valid for the discrete time WDF. Yen [Ref. 15] defines the discrete time auto-Wigner distribution for a sampled time signal $s(t)$, for $0 \leq t < T$, as:

$$WDF_{s,s}(t, \omega) = \frac{1}{T} \sum_{\tau=0}^{T-1} e^{-j\frac{4\pi\omega\tau}{T}} s(t+\tau) s^*(t-\tau) \quad (12)$$

Both equations (11) and (12) are similar. In either case the WDF is basically the Fourier transform of an auto correlation of a signal.

B. WIGNER-VILLE DISTRIBUTION

In 1948, Ville proposed the use of analytic signals in time- frequency representations of real signals [Ref. 16]. An analytic signal is a complex signal which contains both real and imaginary components. The advantage of using an analytic signal is that in the frequency domain the amplitudes of negative frequency components are zero. This satisfies mathematical completeness of the problem by accounting for all frequencies, yet does not limit our practical application since only positive frequency components have a practical interpretation.

An analytic signal may be formed from a real signal by several methods. These methods may be grouped into either a time domain formulation or a frequency domain formulation. In the time domain a typical formulation uses the Hilbert transform and may be expressed as: [Ref. 17]

$$s(t) = s_r(t) + j H\{s_r(t)\} \quad (13)$$

where:

$s(t)$ is the resulting analytic signal which is complex

$s_r(t)$ = real component of a signal

$H\{s_r(t)\}$ = imaginary component of a signal

$H\{s_r(t)\}$ is a Hilbert transform and is defined as:

$$H\{s_r(t)\} = \begin{cases} s_r(t) \times \left[\frac{\sin^2\left(\frac{n\pi}{2}\right)}{\left(\frac{n\pi}{2}\right)} \right] & t \neq 0 \\ 0 & t = 0 \end{cases}$$

In the frequency domain a typical formulation uses the Fast Fourier Transform (FFT) and is expressed as: [Ref. 18]

$$s(t) = FFT^{-1}[S(\omega)] \quad (14)$$

where:

$$S(\omega) = \begin{cases} S_R(\omega) & \omega = 0 \\ 2 S_R(\omega) & \omega = 1, \frac{N}{2} - 1 \\ 0 & otherwise \end{cases}$$

$$S_R(\omega) = FFT [s_r(t)]$$

The distribution resulting from an analytic signal being processed through the Wigner distribution is commonly termed a Wigner-Ville Distribution.

C. PSEUDO WIGNER-VILLE DISTRIBUTION

The Wigner-Ville Distributions of most signals are very complicated and difficult to interpret since the input signals consist of many components. The most annoying characteristic of the Wigner and Wigner-Ville Distribution is the presence of cross terms.

Jones and Parks [Ref. 19] described the production of a cross term for the sum of two signals. The Wigner Distribution of the sum of two signals, $r(t) + s(t)$, is defined as:

$$WDF_{r+s, r+s}(t, \omega) = WDF_{r, r}(t, \omega) + 2 \operatorname{Re}[WDF_{r, s}(t, \omega)] + WDF_{s, s}(t, \omega) \quad (15)$$

The annoying cross term results from the cross-Wigner distribution $WDF_{r, s}$, and is located midway between the auto terms. As the input signals consist of a summation of greater numbers of individual components, the number of cross terms also increases.

There are several approaches for the removal, or deemphasis, of these cross terms. For machinery condition monitoring applications the presence of cross terms in the Wigner-Ville Distribution is not disastrous as long as they can be identified as such. However, they will make the resulting distributions more difficult to interpret. Usually an averaging process is performed in order to produce a more understandable and interpretable representation of the input signal.

There are two methods for making the Wigner Distribution more presentable. Claasen and Mecklenbrauker describe the application of a sliding window in the time domain before calculating the Wigner Distribution [Ref. 12]. The resulting distribution is called a Pseudo Wigner Distribution. A second option is to smooth the Wigner Distribution with a sliding averaging window in the time-frequency plane, sometimes referred to as a Smoothed Wigner Distribution. Since the effect of this second method is essentially the same as the first, the removal of undesired components, the resulting distribution from either method will be called a Pseudo Wigner Distribution. In both cases the result is to deemphasize components arising from calculations and to emphasize deterministic components. Obviously, averaging a Wigner-Ville Distribution will result in a Pseudo Wigner-Ville Distribution.

There has been a fair amount of research as to the optimum smoothing algorithm. References 12, 15, 20, 21, 22, and 23 all discuss various methods available. Due to the fact that the resulting Pseudo Wigner-Ville Distributions were to be used to identify time varying frequency components of machinery signatures, a sliding exponential window in the time-frequency plane is used in this work for the averaging process. This decision is supported by Nuttall's work which demonstrated that smoothing must be applied to both time and frequency [Ref. 23].

III. THE WIGFUN COMPUTER PROGRAMS

A. BRIEF DESCRIPTION

The computer program developed in this research is included as Appendix A. The computer code begins with a summary of the programs, subroutines, and symbols used. The computer code then lists the WIGFUN Fortran programs, is followed by an alphanumeric listing of the subroutines, and is concluded with a listing of the data format conversion program referred to in Appendix B. A flow chart of the WIGFUN programs follows:

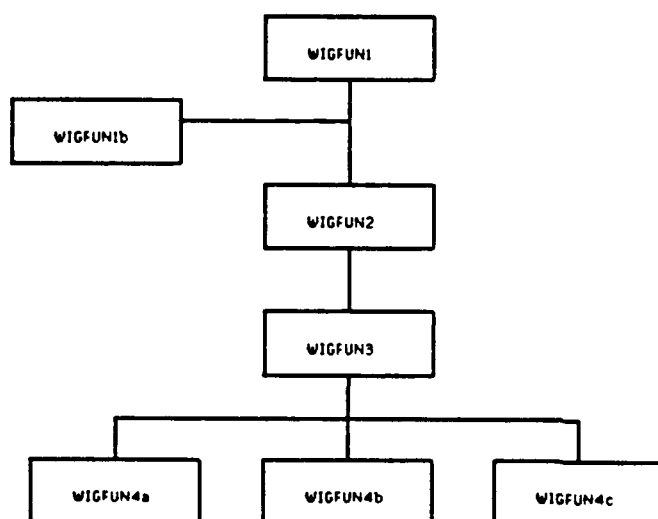


Figure 2. Flow Chart of WIGFUN Computer Programs

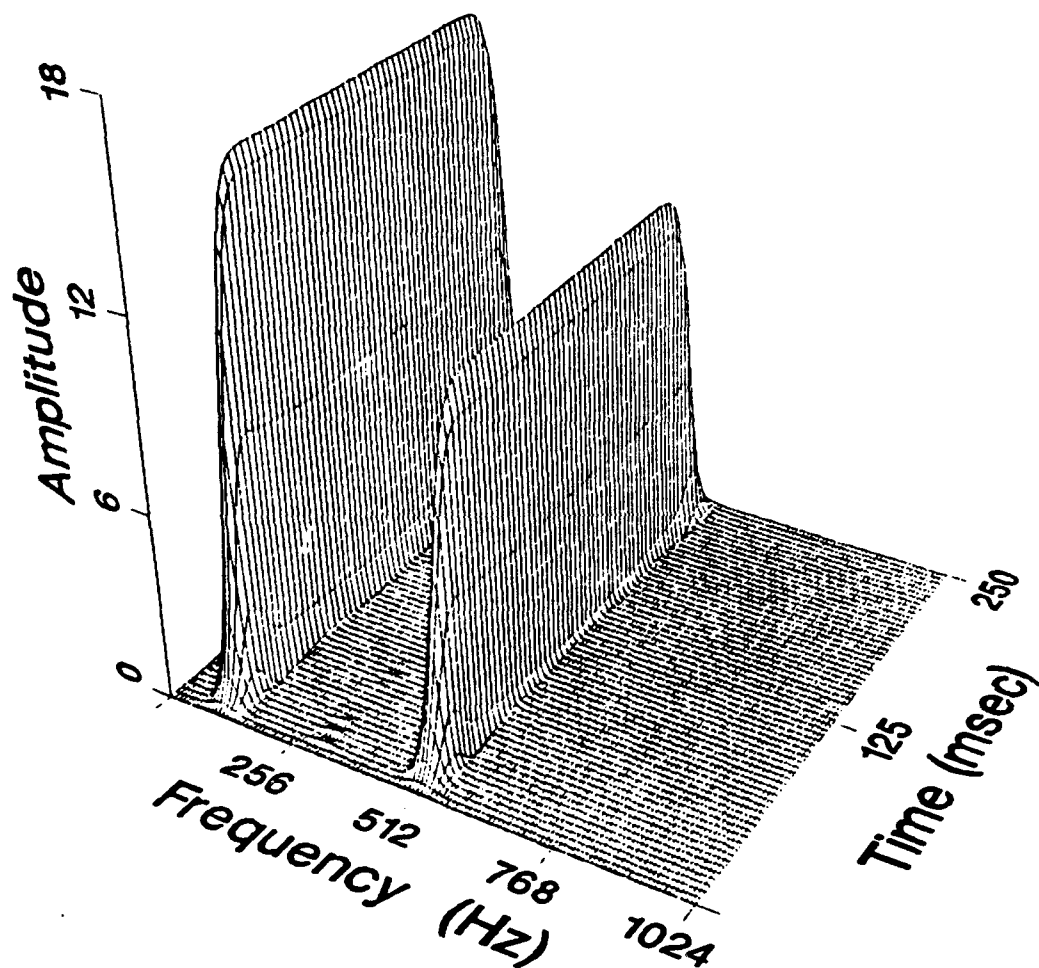
The program is written in Fortran 77 and is designed to be interactive so few, if any, modifications should be required for implementation by other users. The programs may not have been validated for all cases of interest. The 'include' statements in the Fortran programs cause the compiler to include the indicated file at compilation. The included files are individual subroutines for the most part, making the size of the individual programs and subroutines more manageable. The plotting routines all use *CA - DISSPLA*[®] software [Ref. 24].

A description of the process of transferring data from a source to the Digital VAX computer used is provided in Appendix B. The only requirement for the data to be used in the computer program is that it is in the format read in subroutine DATAIN.INCLUDE.

All of the graphs presented in this thesis resulted from analog signals obtained in the laboratory and transferred into the VAX computer using the procedure provided in Appendix B.

Figure 3 on page 11 and Figure 5 on page 13 show two Pseudo Wigner-Ville Distributions. Figure 3 is the result obtained from operating on two sine waves generated by function generators and then added together. It can be clearly seen that the frequency content remains constant over all time, as expected for a sine wave. Figure 4 on page 12 is the time signal input for Figure 3. Figure 5 is a linear chirp which was obtained from a sine wave whose frequency was linearly varied with a triangular wave. This figure demonstrates that the Pseudo Wigner-Ville Distribution is able to represent time varying frequencies. Figure 6 on page 14 is the time signal input for Figure 5. Figure 7 on page 15 shows different viewing angles of the two sine waves seen in Figure 3. The top left view shows all three axes, the top right view looks down the frequency axis, the bottom left view looks down the time axis, and the bottom right view looks down the amplitude axis at a single plane located at one third of the maximum amplitude. These four views provide a good, fast presentation of the distribution. A more detailed contour plot, as in Figure 8 on page 16, gives a better indication of the time varying characteristics of the input signal. A practical machinery monitoring example is presented in chapter 4. A description of the computer program listed in Appendix A and the options available follows.

Pseudo Wigner-Ville Distribution 100 Hz and 500 Hz sin waves

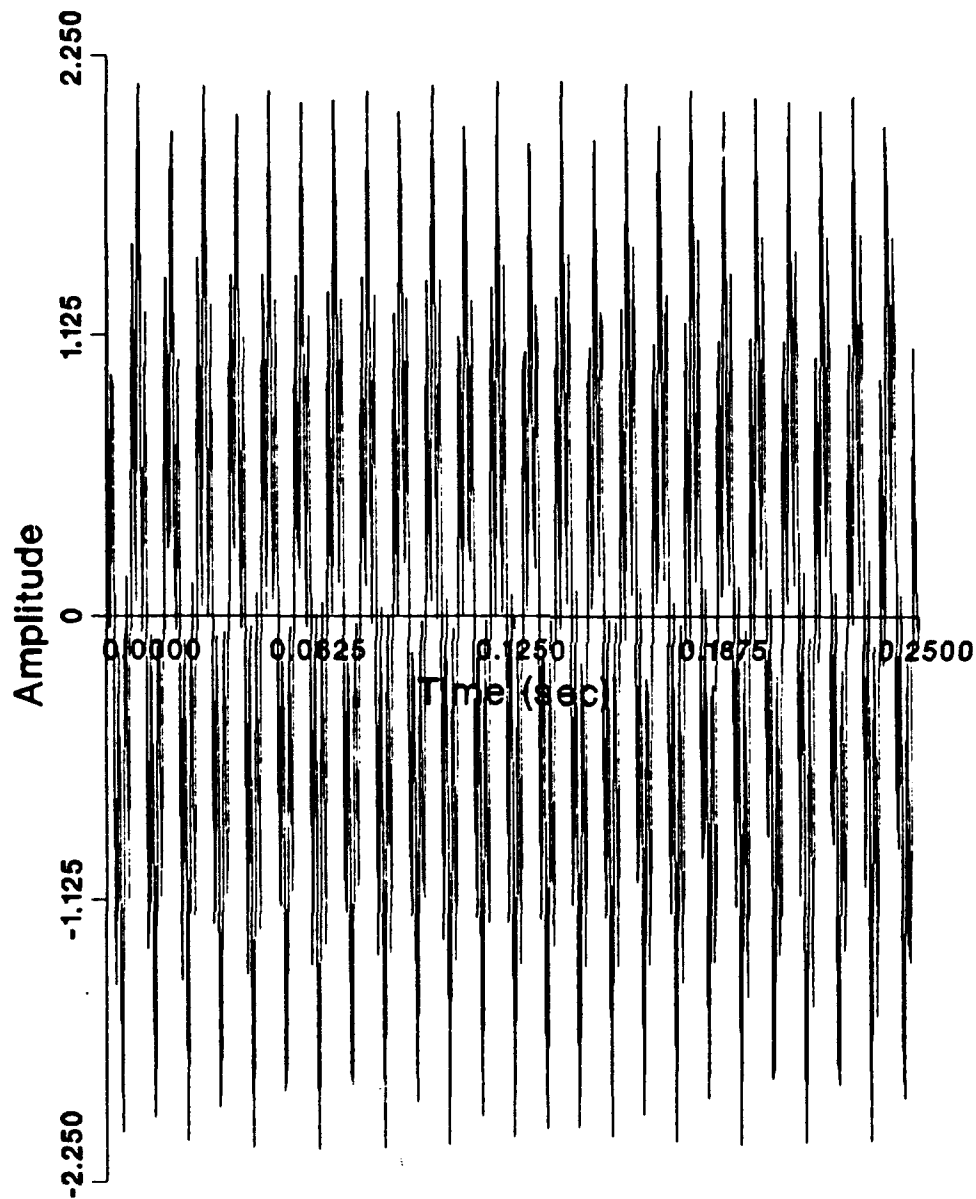


11-APR-1990 13:45:01.97
512 data points
Reduced to 128 x 64
Smoothed 10 x 10

Mean value removed
Time amplified by 0.0
Hamming window time

Figure 3. Pseudo Wigner-Ville Distribution of Two Sine Waves

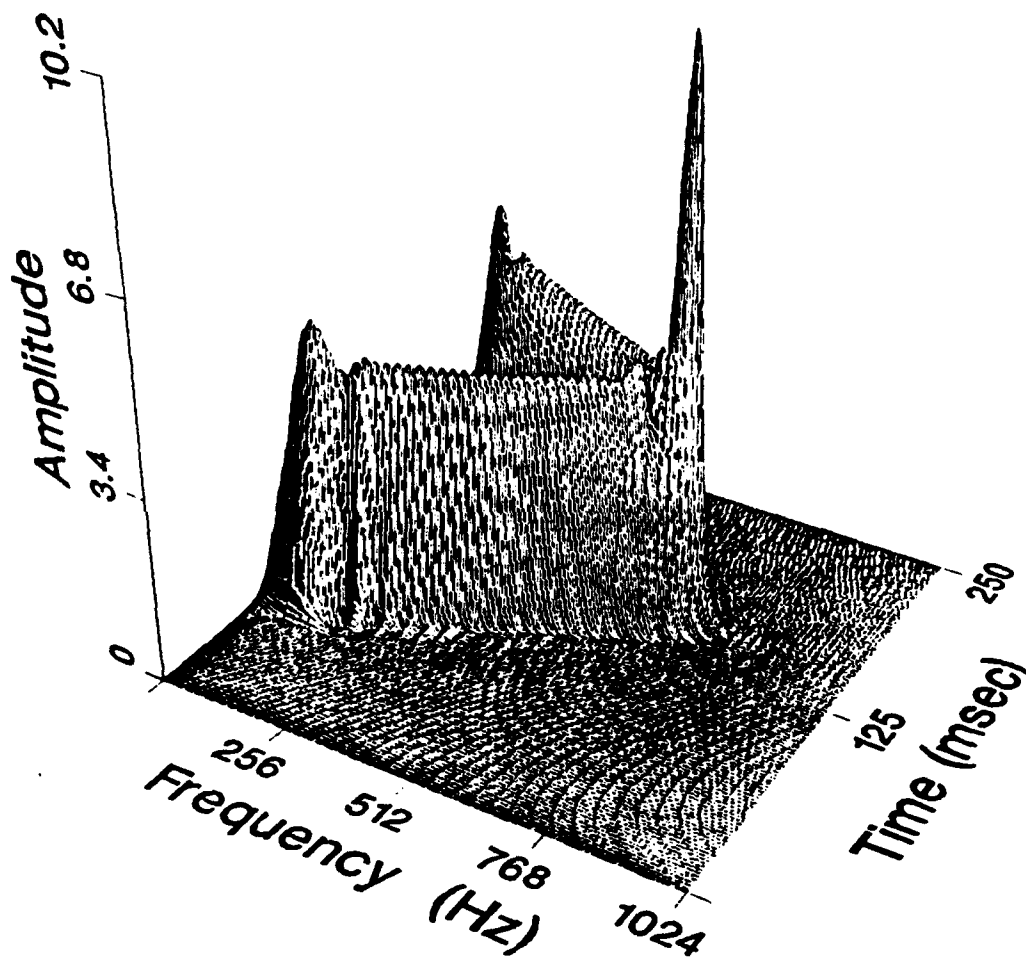
Raw Time Signal 100 Hz and 500 Hz sin waves



11-APR-1990 13:06:37.00

Figure 4. Time Signal for Two Sine Waves

Pseudo Wigner-Ville Distribution Linear Chirp



16-APR-1990 12:17:40.91
512 data points
Reduced to 256 x 128
Smoothed 10 x 10

Mean value removed
Time amplified by 0.0
Hamming window time

Figure 5. Pseudo Wigner-Ville Distribution of a Linear Chirp

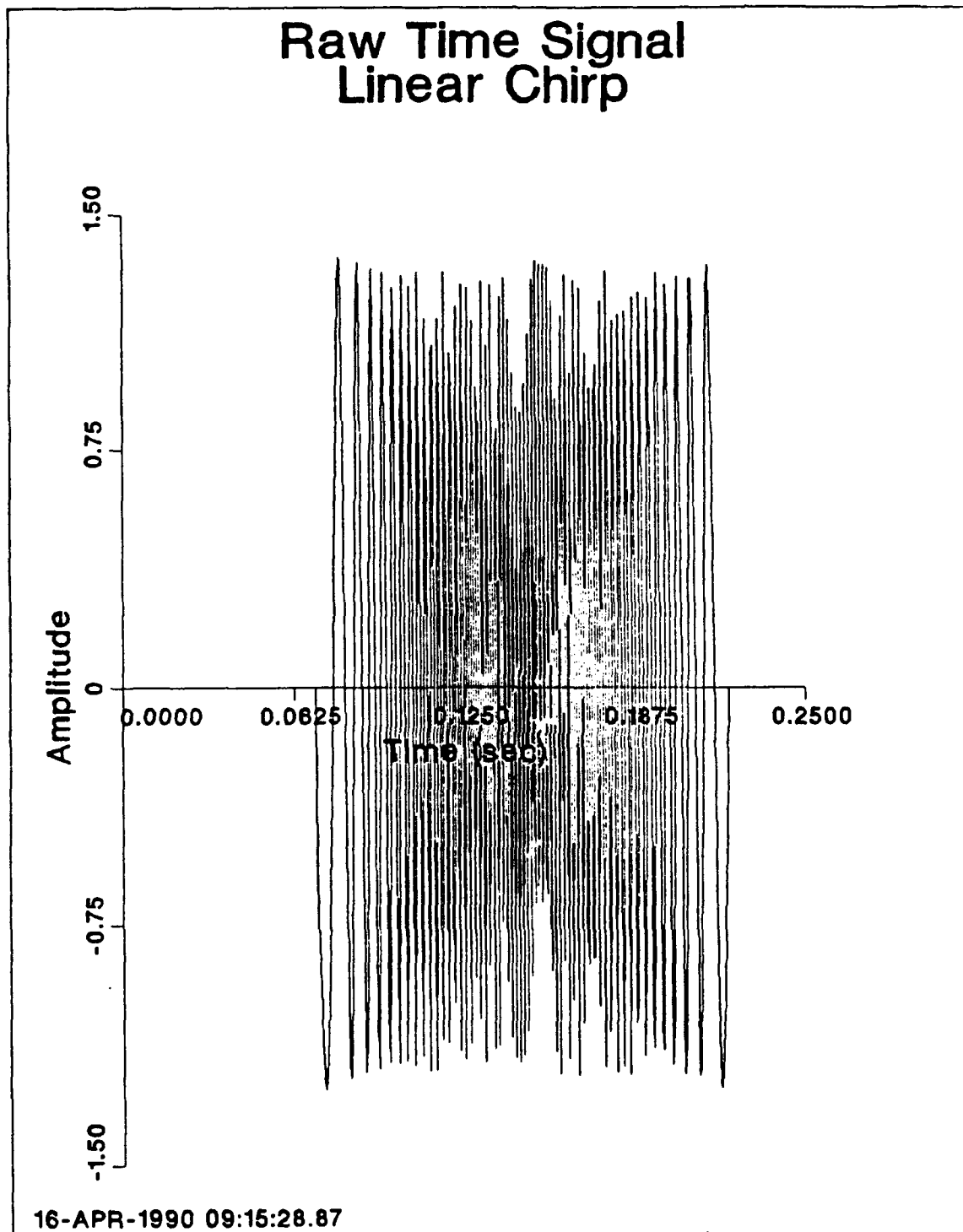
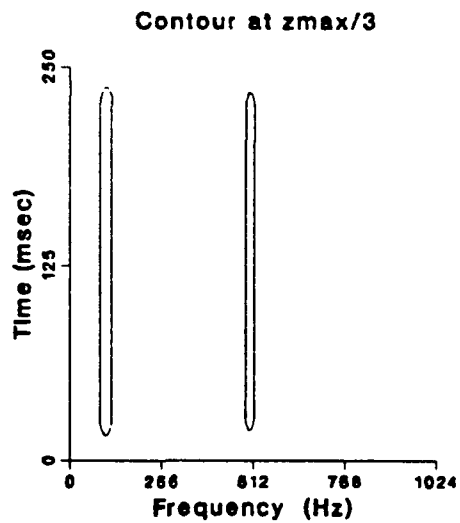
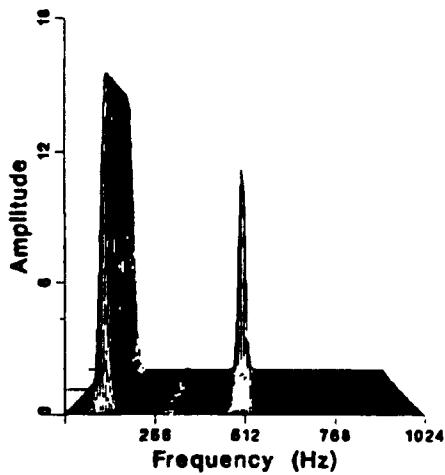
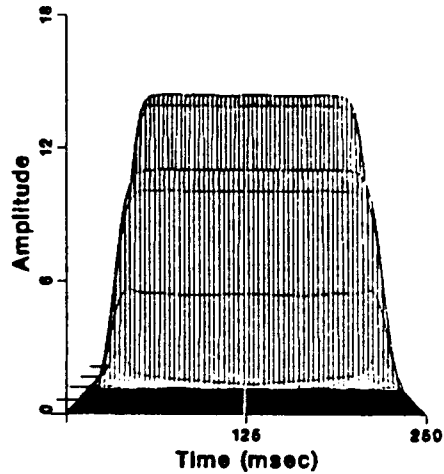
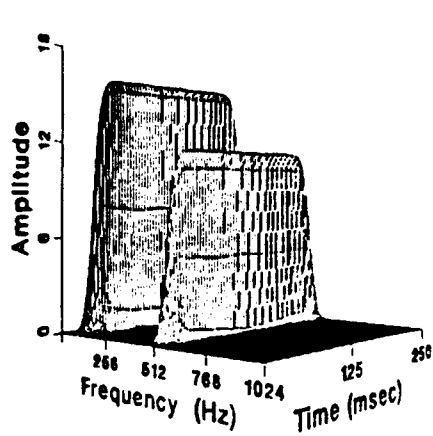


Figure 6. Time Signal for Linear Chirp

Pseudo Wigner-Ville Distribution 100 Hz and 500 Hz sin waves



11-APR-1990 13:38:30.33
512 data points
Reduced to 128 x 64
Smoothed 10 x 10

Mean value removed
Time amplified by 0.0
Hamming window time

Figure 7. Different Viewing Perspectives

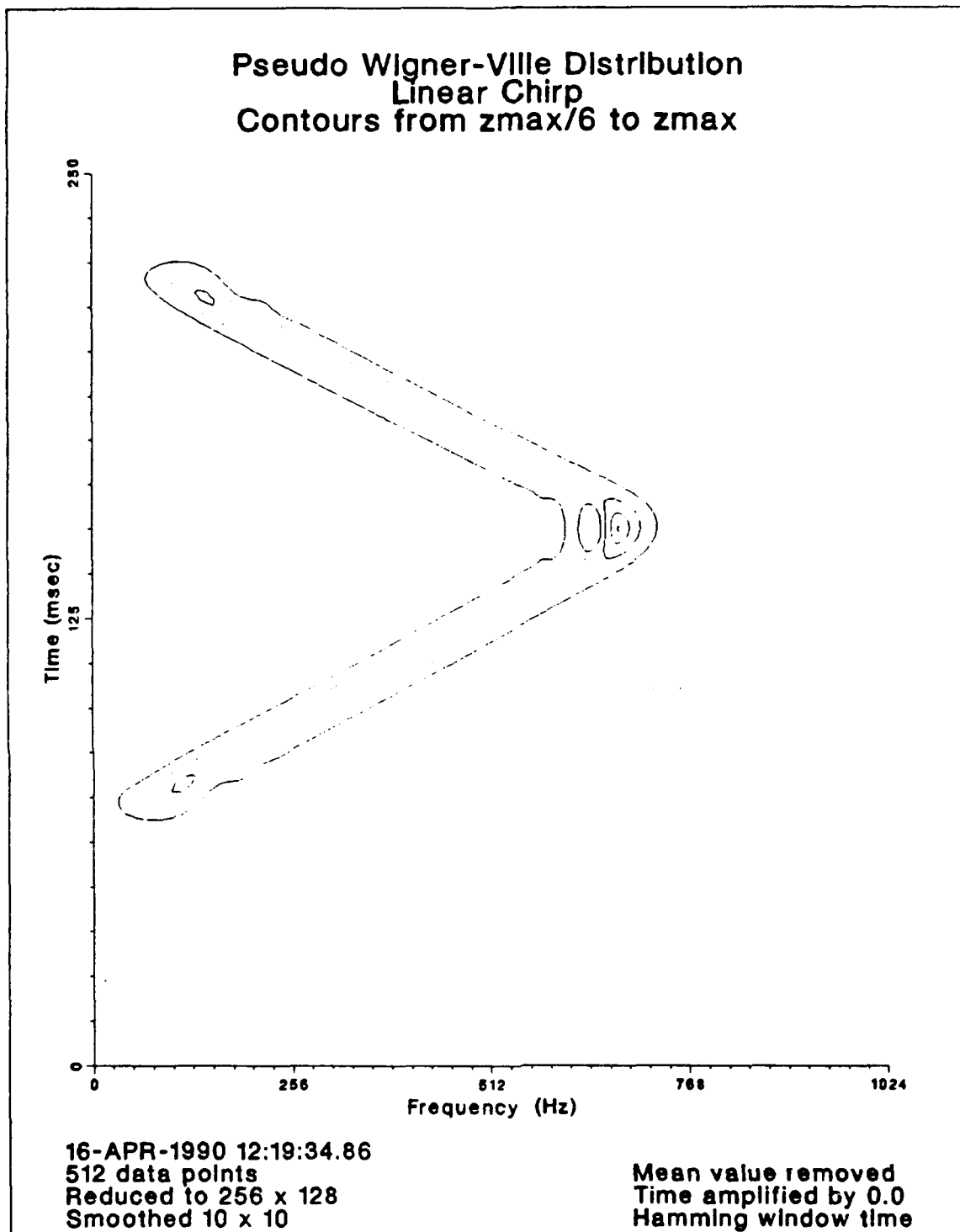


Figure 8. Detailed Contour Plot of Linear Chirp

B. LIMITING DECISIONS

As can be seen in Figure 2 on page 9, the first Fortran program is WIGFUN1. WIGFUN1 reads in the raw time data, preprocesses it, and outputs an analytic data file. There are several data preprocessing options available within WIGFUN1 and each will be discussed. However, prior to preprocessing, several limiting decisions must be made.

The first decision is the number of data points to use. The maximum number of data points which may be processed is set by the array sizes dimensioned throughout the computer code. As printed the listing is set for 512 data points. If more data points are required then larger arrays should be dimensioned. Consequently, more computer storage space is required and the run time is longer. It was found that for initial applications 512 data points were sufficient. Due to the use of a Fast Fourier Transform subroutine, the number of data points must be a power of 2 (128, 256, 512, 1024, etc.). If the data file to be used does not have as many data points as were chosen, the program will automatically pad the remaining points with zeros. The program will interpret this as an input signal value of zero.

The second limiting decision which must be made is what time step size (Δt) to use. Initially the program reads the data and calculates the average step size. From this average step size the maximum time, maximum frequency, and frequency resolution are calculated. These quantities are related as follows:

$$mtime = dp \times \Delta t \quad (16)$$

$$\Delta f = \frac{1}{4 \times mtime} \quad (17)$$

$$mfreq = 2 \times dp \times \Delta f \quad (18)$$

where:

$mtime$ = maximum time

$mfreq$ = maximum frequency

dp = the number of data points

Δt = time step or time resolution

Δf = frequency step or frequency resolution

In machinery monitoring applications usually two frequency spans are used, a broadband and a narrow band measurement. These displays may be obtained by either changing the filters before digitizing the data or by varying the time step size in the computer program. A third possibility, which has not been implemented but could be,

is the inclusion of a digital filter in the computer program. By selecting the smallest possible Δt the largest frequency span is obtained. The computer program will narrow the frequency span and focus on the lower frequencies by using a larger Δt . The varying size of the Δt is achieved by using every other data point or larger multiples of data points. Obviously the maximum frequency possible is determined by the Δt at digitization. Sometimes the Δt after digitization is not constant and the program allows for this by calculating an average Δt and using this average value as the actual Δt .

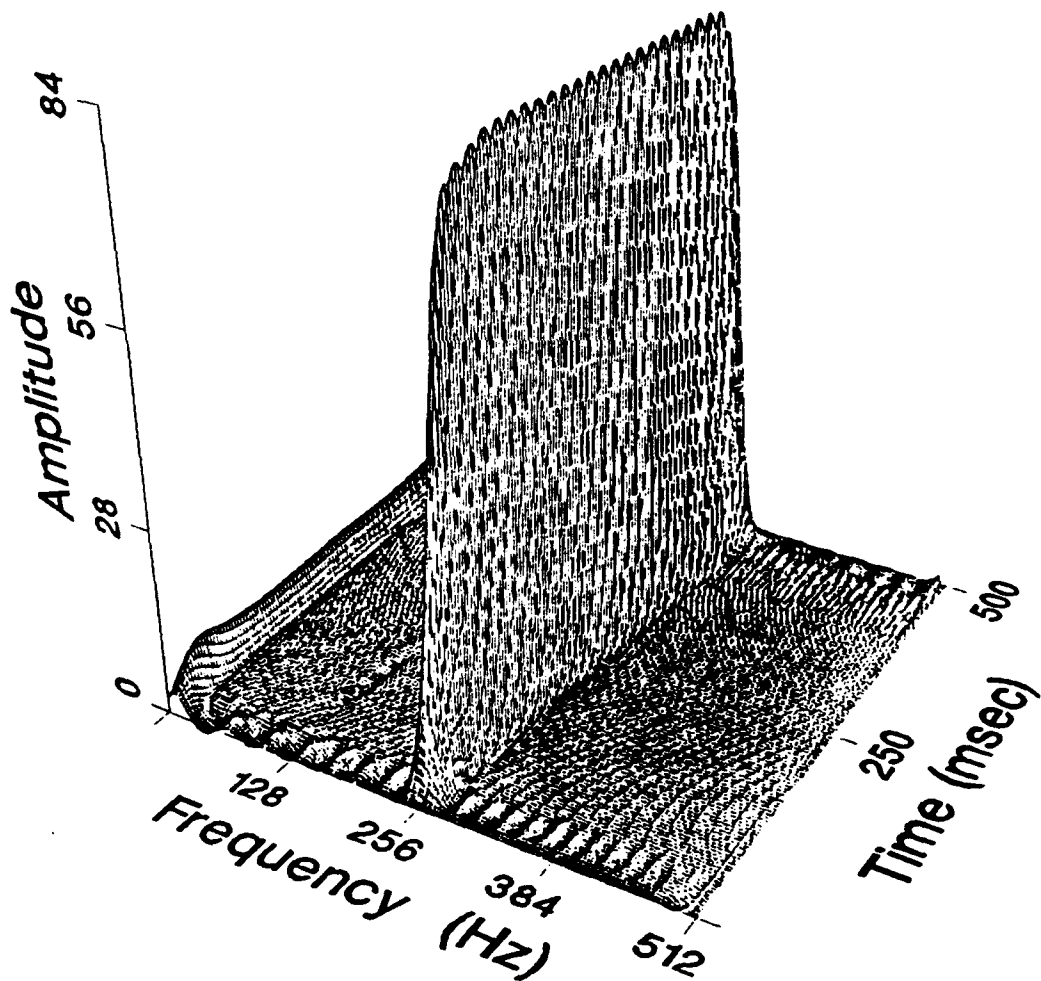
C. WIGFUN1

As mentioned earlier, WIGFUN1 is a time domain preprocessing program. It reads in the raw time data and outputs an analytic data file. Plotting subroutines are included in order to output various curves throughout the process. Before the analytic signal is calculated there are several options which may be exercised. The implications and effects of each are discussed below.

1. Effect of a Mean Value

Figure 9 on page 19 and Figure 10 on page 20 demonstrate the effect of a mean value in the time domain. Figure 9 is a sine wave whose frequencies have been slowly varied and which has a mean value. As can be seen, the mean value in the time domain appears as a DC (0 Hz) component in the frequency domain. Figure 10 shows the same linear chirp which has had the mean value removed. As expected, the DC component has been eliminated. Figure 11 on page 21 is the input time signal for the linear chirp shown in Figure 9 and Figure 10.

Pseudo Wigner-Ville Distribution Linear Chirp

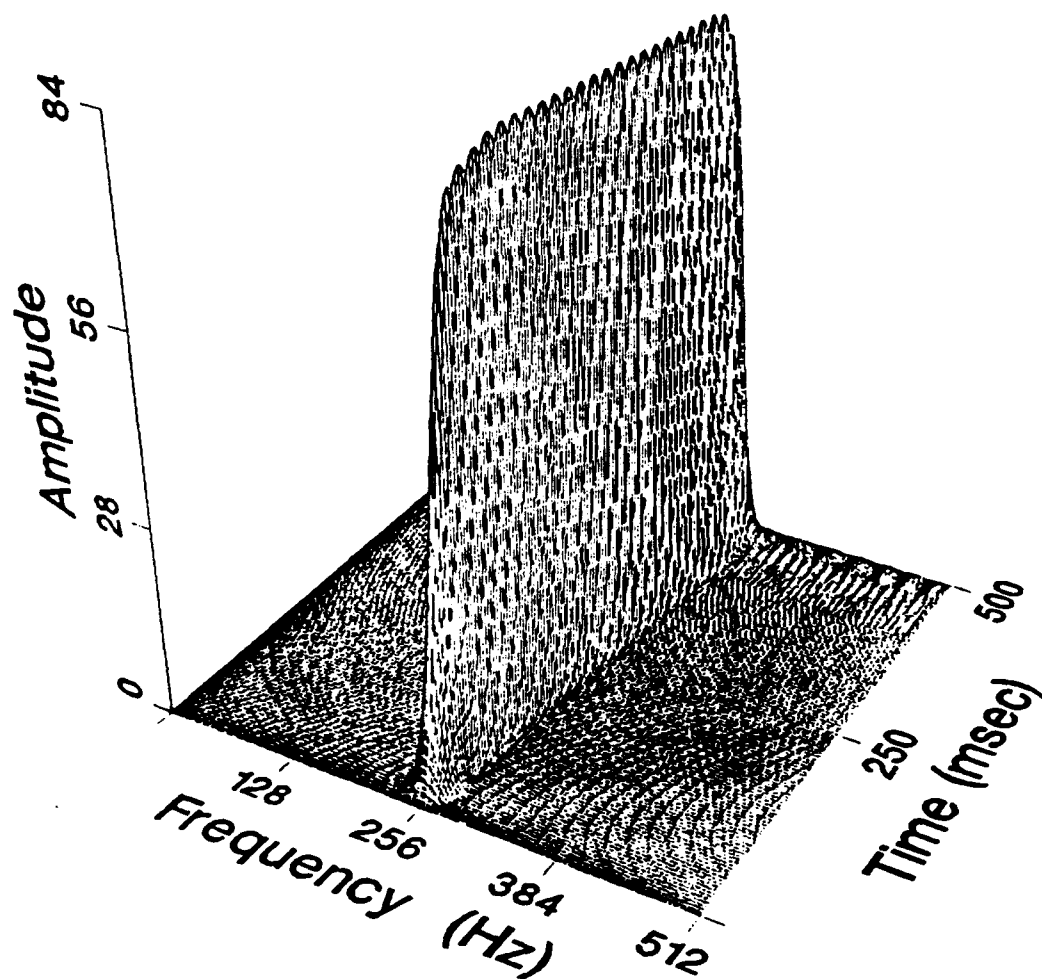


26-MAR-1990 20:18:40.59
512 data points
Reduced to 256 x 128
Smoothed 10 x 10

Mean value not removed
Time amplified by 0.0
Hamming window time

Figure 9. Linear Chirp with Mean Value in the Time Domain

Pseudo Wigner-Ville Distribution Linear Chirp



27-MAR-1990 08:05:56.40
512 data points
Reduced to 256 x 128
Smoothed 10 x 10

Mean value removed
Time amplified by 0.0
Hamming window time

Figure 10. Linear Chirp with Mean Value Removed from the Time Domain

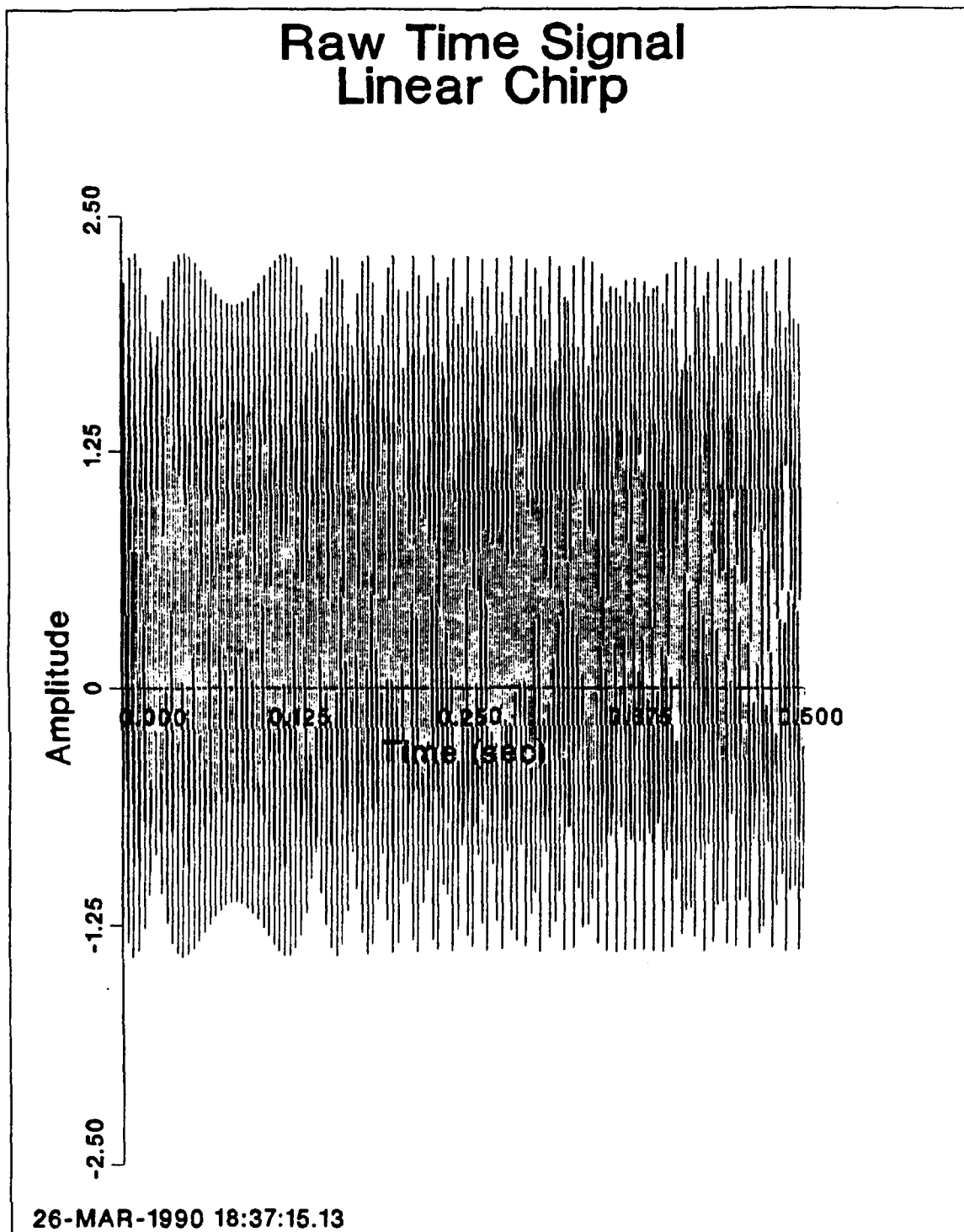
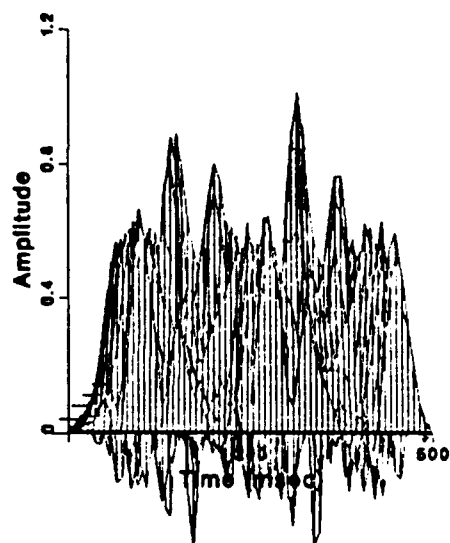
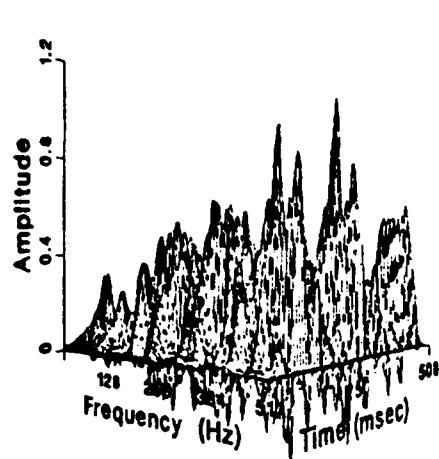


Figure 11. Time Signal for Linear Chirp

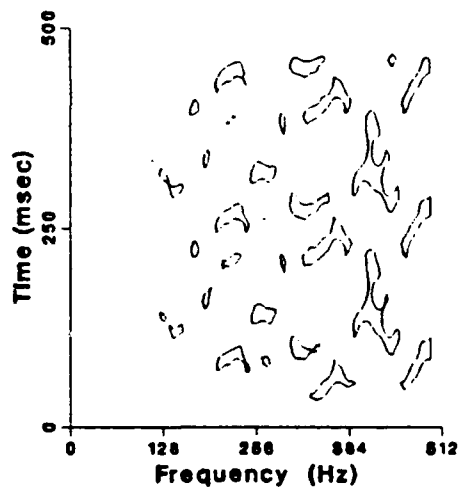
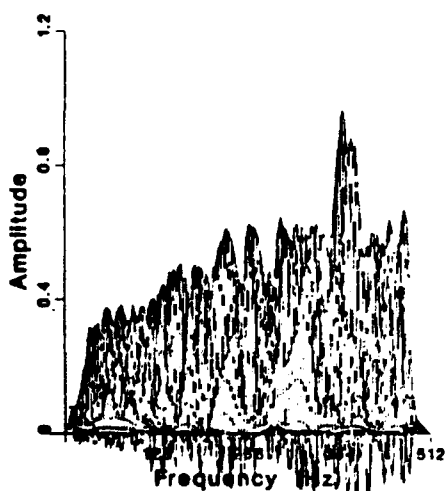
2. Effect of Amplification of Time Signal Amplitude

A subroutine has been included which amplifies the amplitude of the input signal in the time domain. When this is used it definitely alters the energy distribution representation. The only other effect which this has on the Pseudo Wigner-Ville Distribution is that it serves as a constant multiplier, varying the amplitudes of the distribution. Figure 12 on page 23 and Figure 13 on page 24 depict a 400 Hz sine wave in noise which demonstrates this effect. Figure 14 on page 25 is the input time signal which produced Figure 12 and Figure 13.

Pseudo Wigner-Ville Distribution 400 Hz sin wave in noise



Contour at $z_{max}/3$

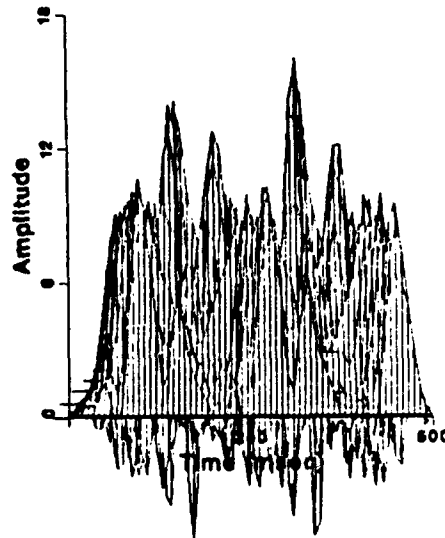
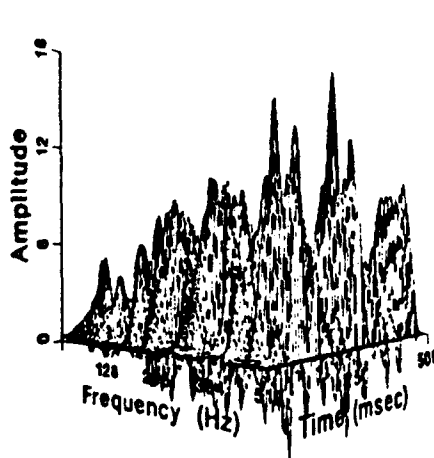


3-APR-1990 12:21:08.24
512 data points
Reduced to 128 x 64
Smoothed 10 x 10

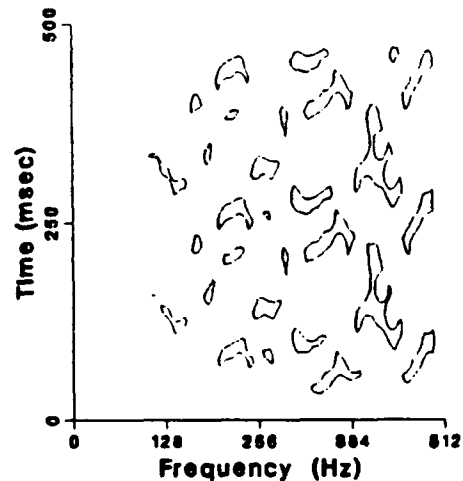
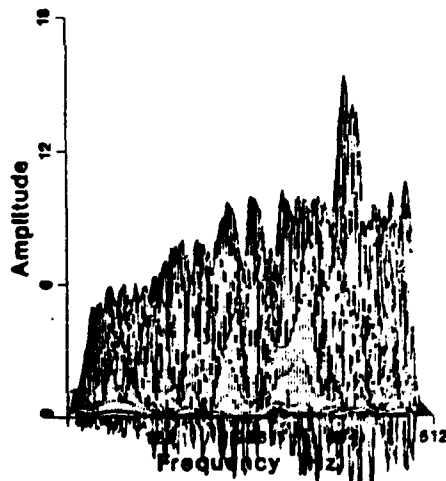
Mean value removed
Time amplified by 0.0
Hamming window time

Figure 12. No Time Signal Amplitude Amplification

Pseudo Wigner-Ville Distribution 400 Hz sin wave in noise



Contour at $z_{max}/3$

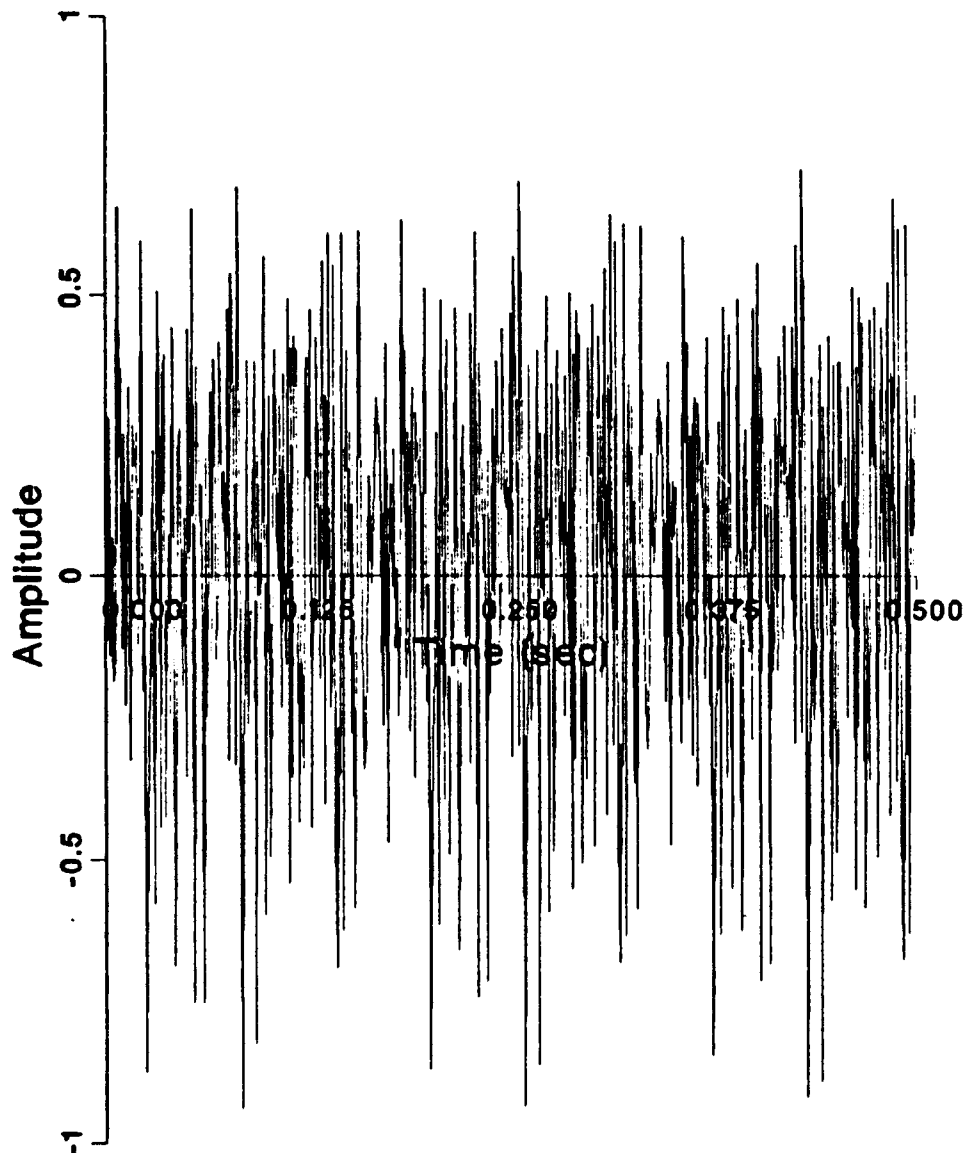


3-APR-1990 13:05:58.85
512 data points
Reduced to 128 x 64
Smoothed 10 x 10

Mean value removed
Time amplified by 4.0
Hamming window time

Figure 13. Time Signal Amplitude Amplification of 4.0

Raw Time Signal 400 Hz sin wave in noise



3-APR-1990 11:49:31.98

Figure 14. Time Signal for 400 Hz Sine Wave in Noise

3. Effect of Time Signal Windows

A subroutine has been included which will window the time domain signal. The reason for windowing the signal in the time domain is to bring the starting and finishing time values of the signal to zero. There are two windows available, a modified Hamming window and a Hanning window. A Generalized Hamming window is defined as: [Ref. 25]

$$window(t) = \begin{cases} \alpha + (1 - \alpha) \cos\left(\frac{\pi t}{T}\right) & -T \leq t \leq T \\ 0 & \text{otherwise} \end{cases} \quad (19)$$

A Hanning window is defined as: [Ref. 26]

$$window(t) = \begin{cases} \frac{1}{2} \left(1 - \cos \frac{2\pi t}{T}\right) & 0 \leq t \leq T \\ 0 & \text{otherwise} \end{cases} \quad (20)$$

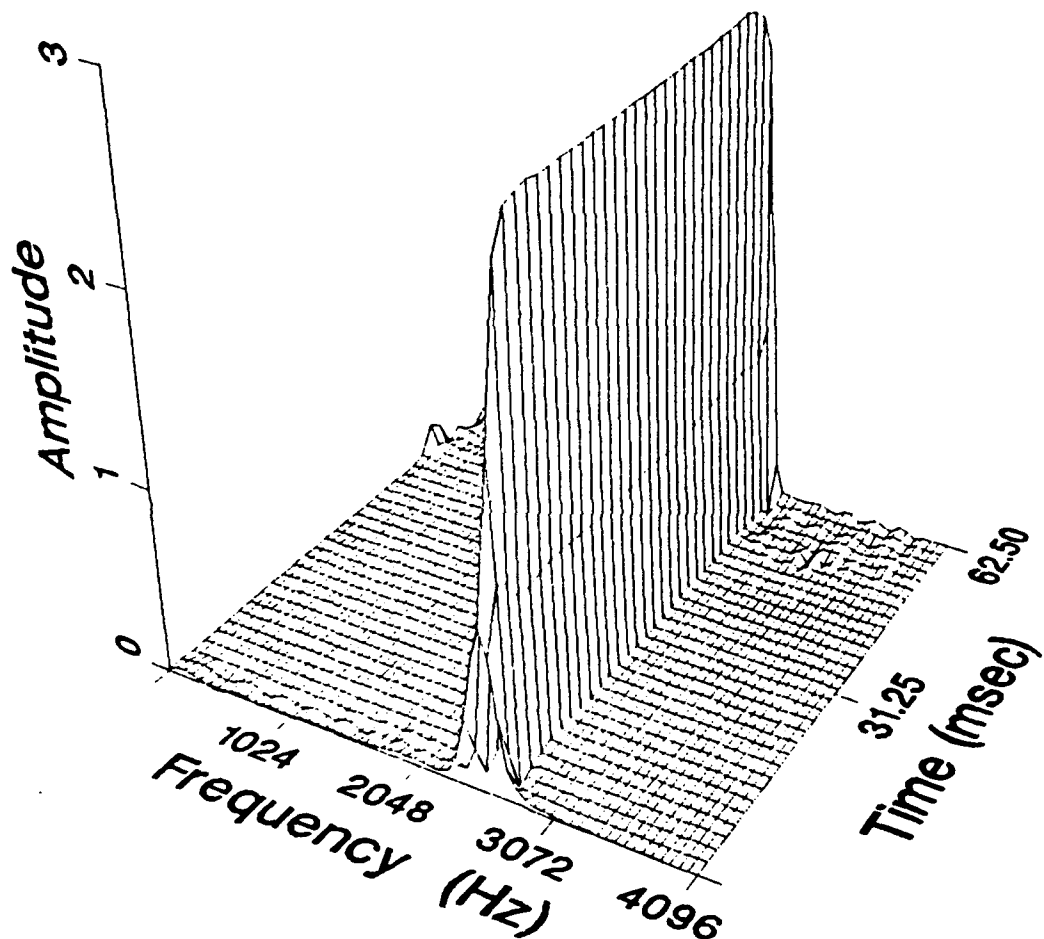
The modified Hamming window used applied a cosine weighting function to the beginning and ending of the input time record and did not vary the amplitudes inbetween. The equation used was:

$$window(t) = \begin{cases} \frac{1}{2} \left(1 - \cos \frac{\pi t}{0.1T}\right) & 0 \leq t \leq 0.1T \\ 1 & 0.1T < t < 0.9T \\ \frac{1}{2} \left(1 - \cos \frac{\pi(T-t)}{0.1T}\right) & 0.9T \leq t \leq T \\ 0 & \text{otherwise} \end{cases} \quad (21)$$

A modified Hamming window is the preferable window since it alters the amplitude of fewer data points than a Hanning window. Figure 15 on page 28 shows a Pseudo Wigner-Ville Distribution of a 2500 Hz sine wave with no window, Figure 16 on page 29 is the input time domain signal. Figure 17 on page 30 shows a Pseudo Wigner-Ville Distribution of a 2500 Hz sine wave with a Hanning window applied in the time domain. Figure 18 on page 31 is the modified time domain signal. Figure 19 on page 32 shows a Pseudo Wigner-Ville Distribution of a 2500 Hz sine wave with the modified Hamming

window applied in the time domain, Figure 20 on page 33 is the modified time domain signal. In Figure 15 it can be seen that with no window the edges of the graph at the start and finish times do not quite reach a zero value, Figure 17 shows that the Hanning window drastically changes the distribution, and Figure 19 shows that the modified Hamming window provides a neater distribution.

Pseudo Wigner-Ville Distribution 2500 Hz sin wave

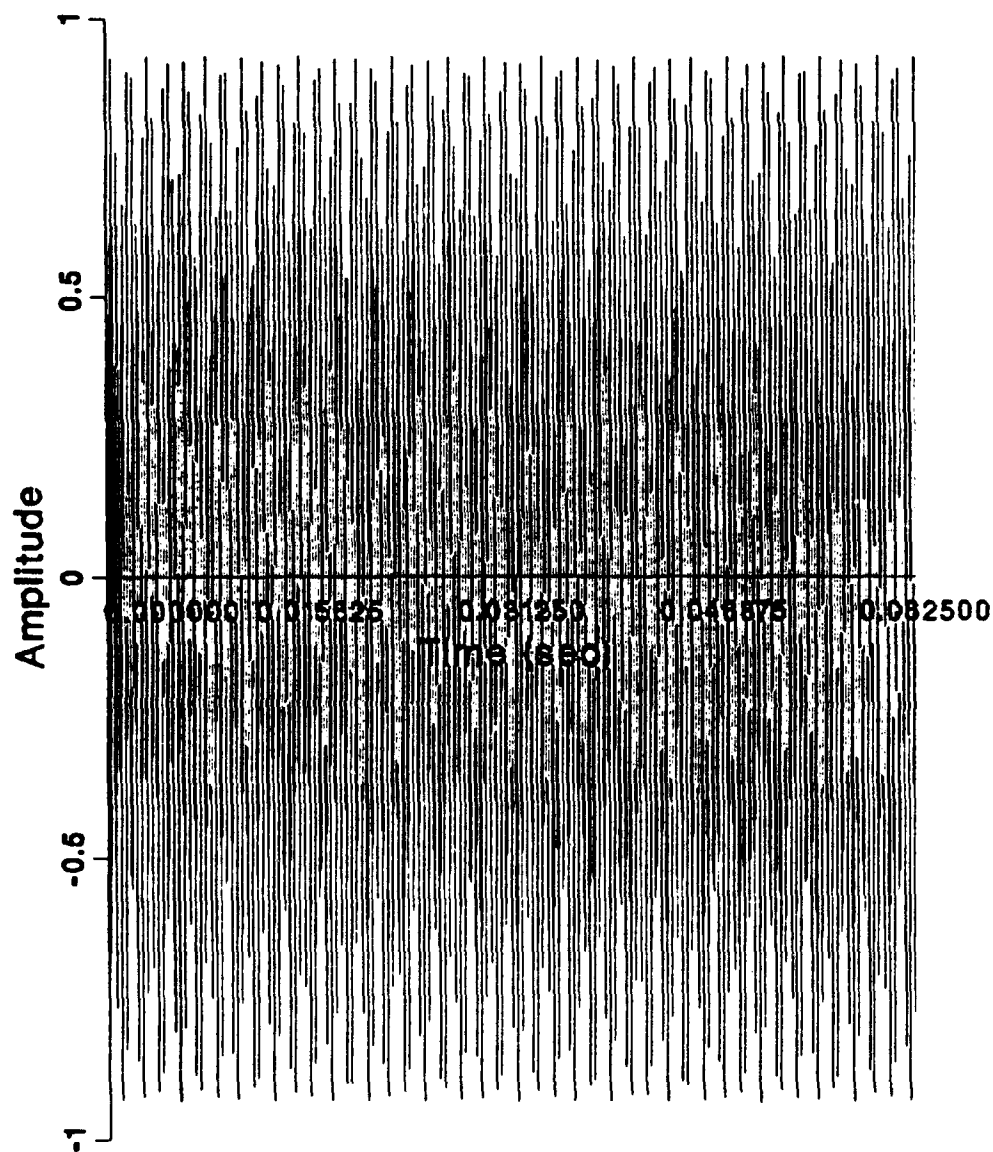


2-APR-1990 18:20:00.51
512 data points
Reduced to 64 x 32
Smoothed 10 x 10

Mean value not removed
Time amplified by 0.0
No window time

Figure 15. Pseudo Wigner-Ville Distribution with No Window

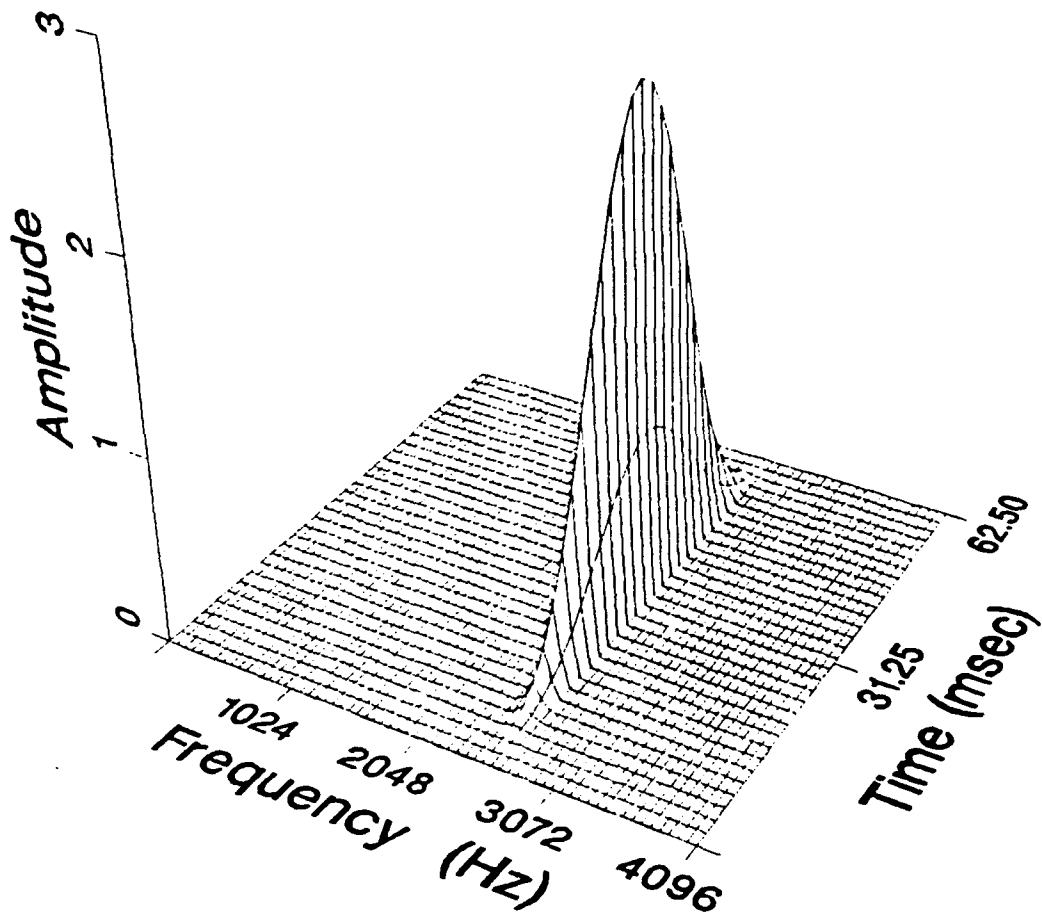
Raw Time Signal 2500 Hz sin wave



2-APR-1990 12:09:07.96

Figure 16. Input Time Signal

Pseudo Wigner-Ville Distribution 2500 Hz sin wave



2-APR-1990 18:31:35.24
512 data points
Reduced to 64 x 32
Smoothed 10 x 10

Mean value not removed
Time amplified by 0.0
Hanning window time

Figure 17. Pseudo Wigner-Ville Distribution with Hanning Window

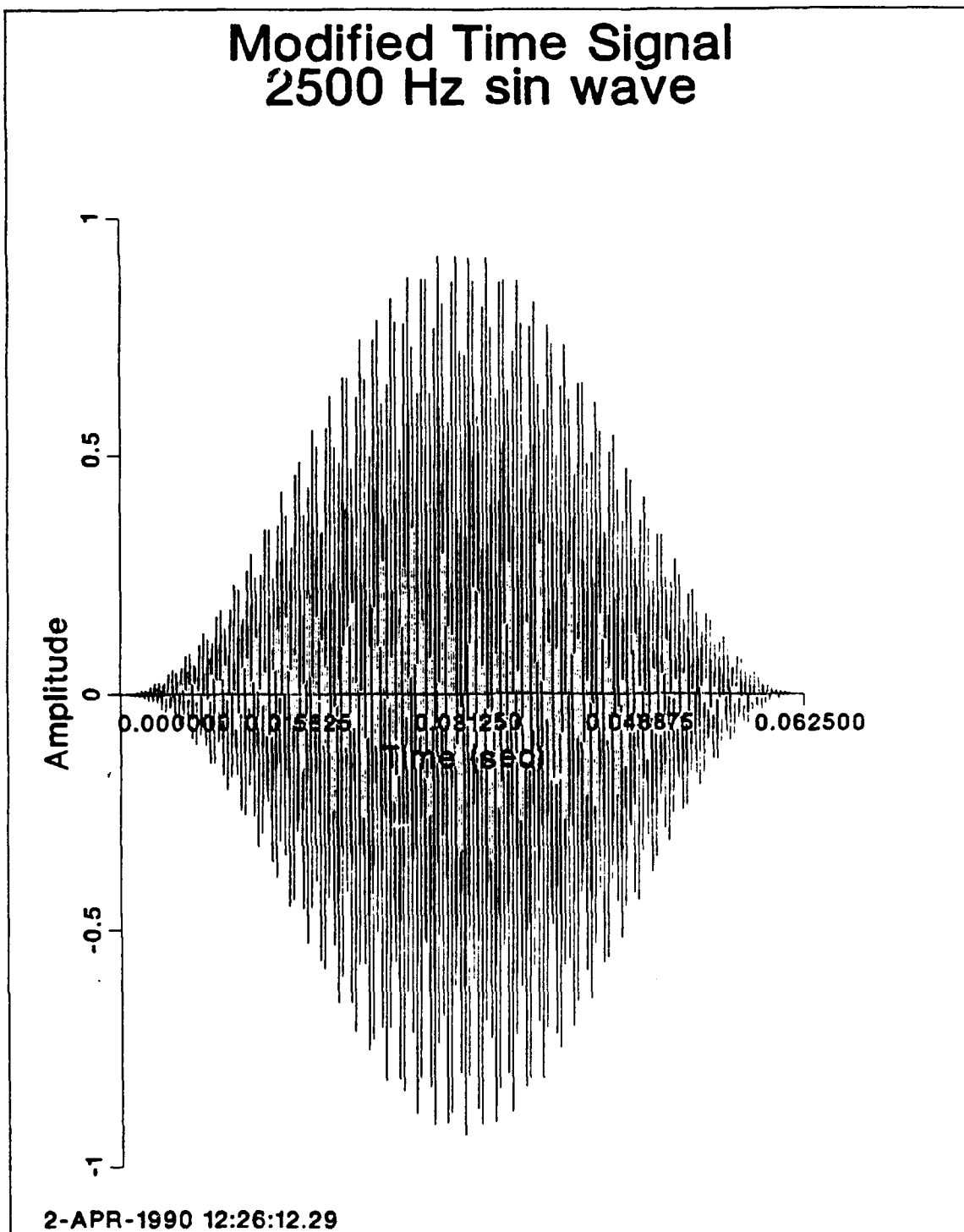
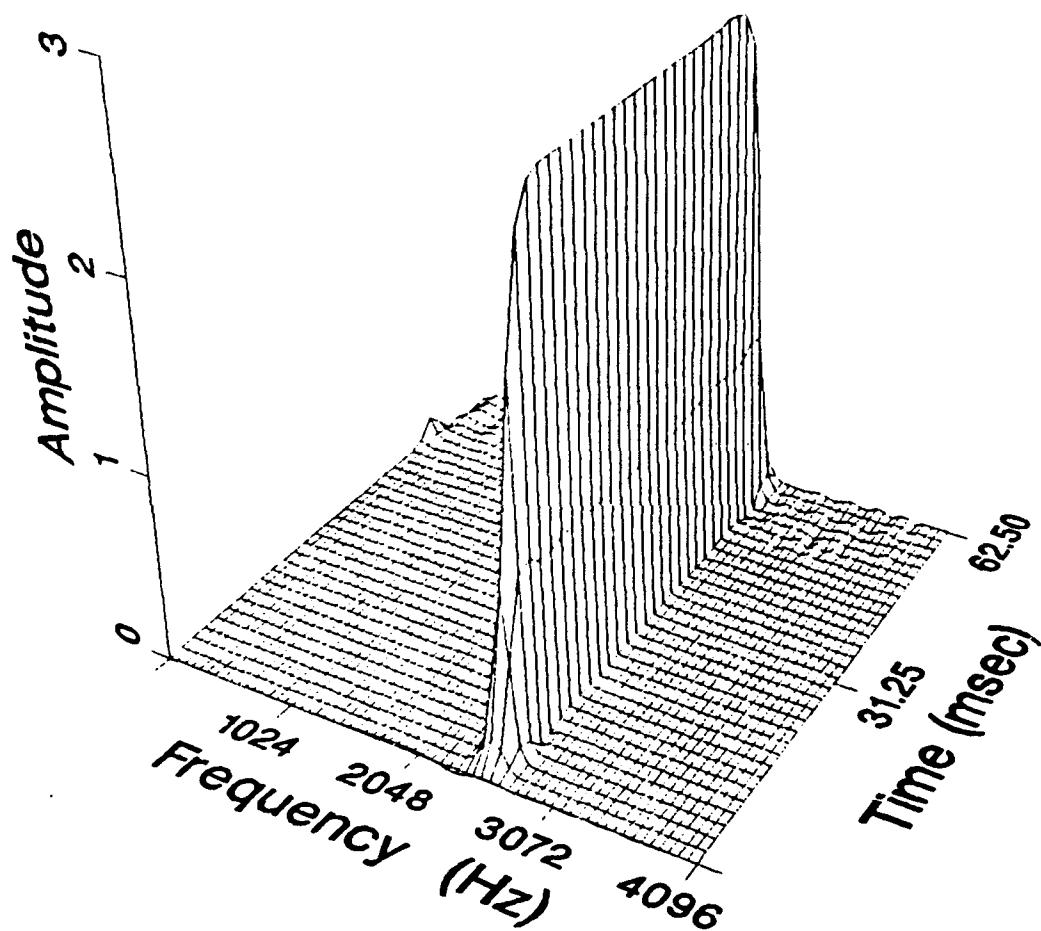


Figure 18. Input Time Signal Modified by Hanning Window

Pseudo Wigner-Ville Distribution 2500 Hz sin wave



2-APR-1990 18:26:44.33
512 data points
Reduced to 64 x 32
Smoothed 10 x 10

Mean value not removed
Time amplified by 0.0
Hamming window time

Figure 19. Pseudo Wigner-Ville Distribution with Modified Hamming Window

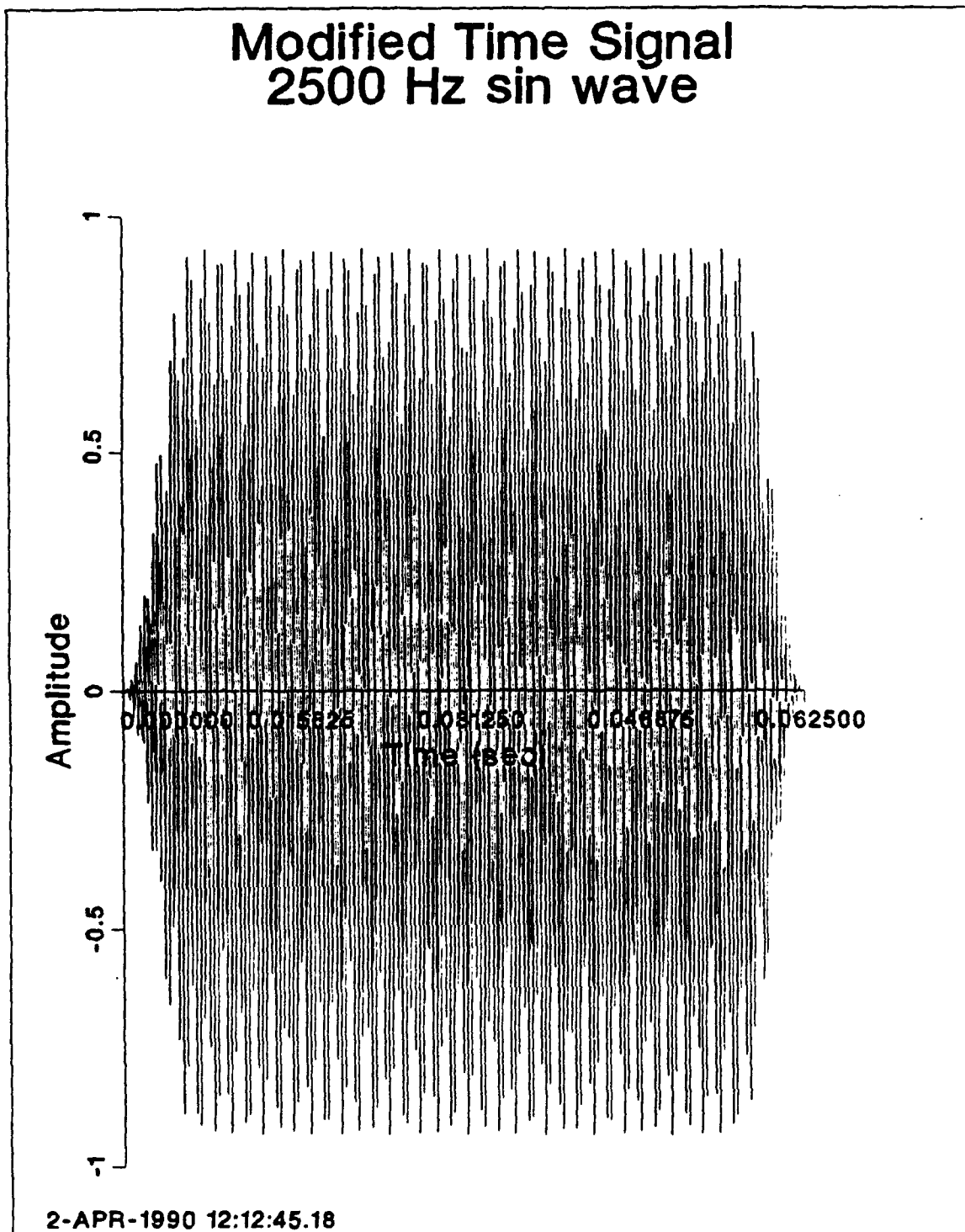
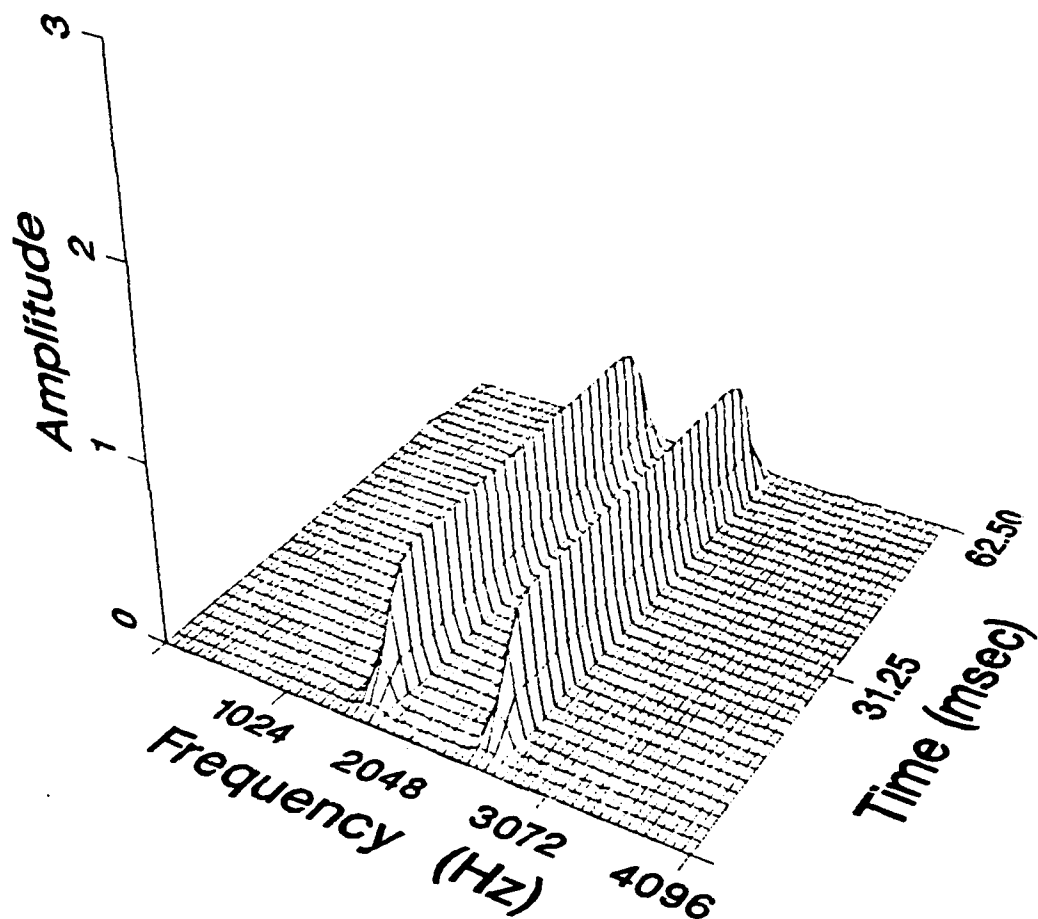


Figure 20. Input Time Signal Modified by Modified Hamming Window

4. Effect of Making a Real Signal Analytic

As discussed in chapter 2, an analytic signal has no negative frequency components. By eliminating these negative frequency components the effects of aliasing are greatly reduced. Figure 21 on page 35 shows the aliasing problem which results when a real signal is not made analytic, Figure 16 on page 29 is the input 2500 Hz sine wave. Figure 22 on page 36 shows a time signal which has been made analytic and has had a modified Hamming window applied. It should be noted that the window was applied before the signal was made analytic, thereby saving some computation time.

Pseudo Wigner Distribution 2500 Hz sin wave



2-APR-1990 18:22:21.07
512 data points
Reduced to 64 x 32
Smoothed 10 x 10

Mean value not removed
Time amplified by 0.0
Hamming window time

Figure 21. Pseudo Wigner Distribution

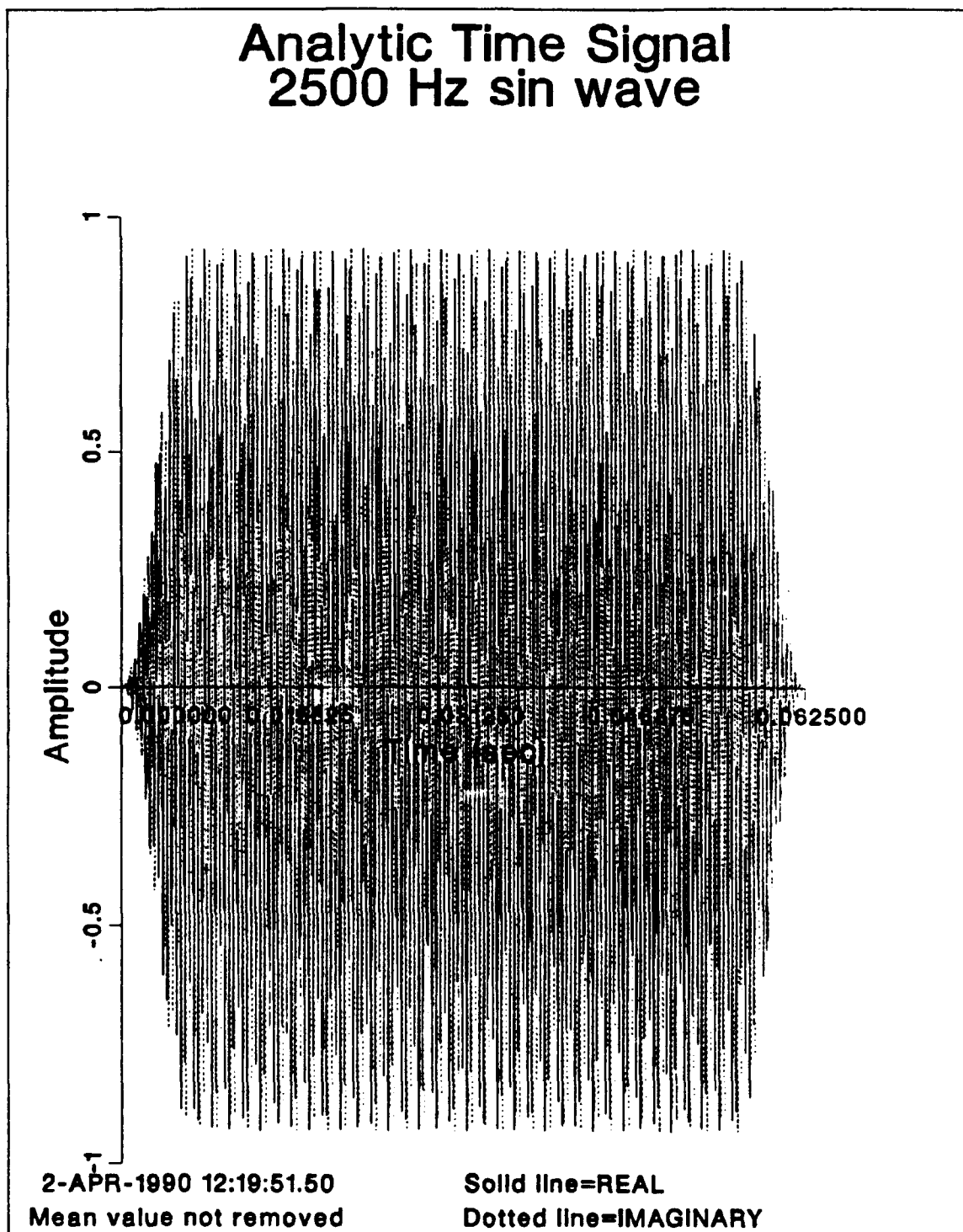


Figure 22. Analytic Time Signal

D. WIGFUN1B

WIGFUN1b is simply a program for plotting the analytic time signal. It is set off by itself in Figure 2 on page 9 since it is not required to be run, but is included as an added convenience.

E. WIGFUN2

WIGFUN2 reads in the analytic time signal from a file and calculates the Wigner-Ville Distribution. The Wigner-Ville Distribution is set equal to the real part resulting from the complex Fourier transform of the calculated auto correlation. The algorithm used is based on one written by Wahl and Bolton [Ref. 8] and can be expressed as:

$$WDF_{s,s}(i\Delta\omega, j\Delta t) = \sum_{j=1}^N \text{Re}\{FFT[2\Delta t \text{ CORR}(i)]\} \quad (22)$$

where:

Re = Real part

$\text{CORR}(i)$ = The complex auto-correlation

$$\text{CORR}(i) = s(j+i-1)s'(j-i+1) \quad 1 \leq i \leq N \text{ and } j < i$$

$$\text{CORR}(i) = 0 \quad 1 \leq i \leq N \text{ and } j \geq i$$

$$\text{CORR}(2N-i+2) = \text{CORR}'(i) \quad N < i \leq 2N$$

1. Effect of Changing the Definition of the Wigner Distribution

As shown in chapter 2, there are several different definitions possible for representing the Wigner Distribution. The computer program calculates the auto correlation of the signal in subroutine CORR.INCLUDE and then takes the complex FFT of the result. In subroutine WIGH.INCLUDE the WDF was set equal to the real part resulting from the FFT. Other possibilities include setting the WDF equal to the imaginary part, setting the WDF equal to the square of the real part, or setting the WDF equal to the square root of the sum of the squares of the real and imaginary parts. Accurate representations of known signals were obtained using just the real part so this was the algorithm used. Additional research could be conducted in order to investigate other options.

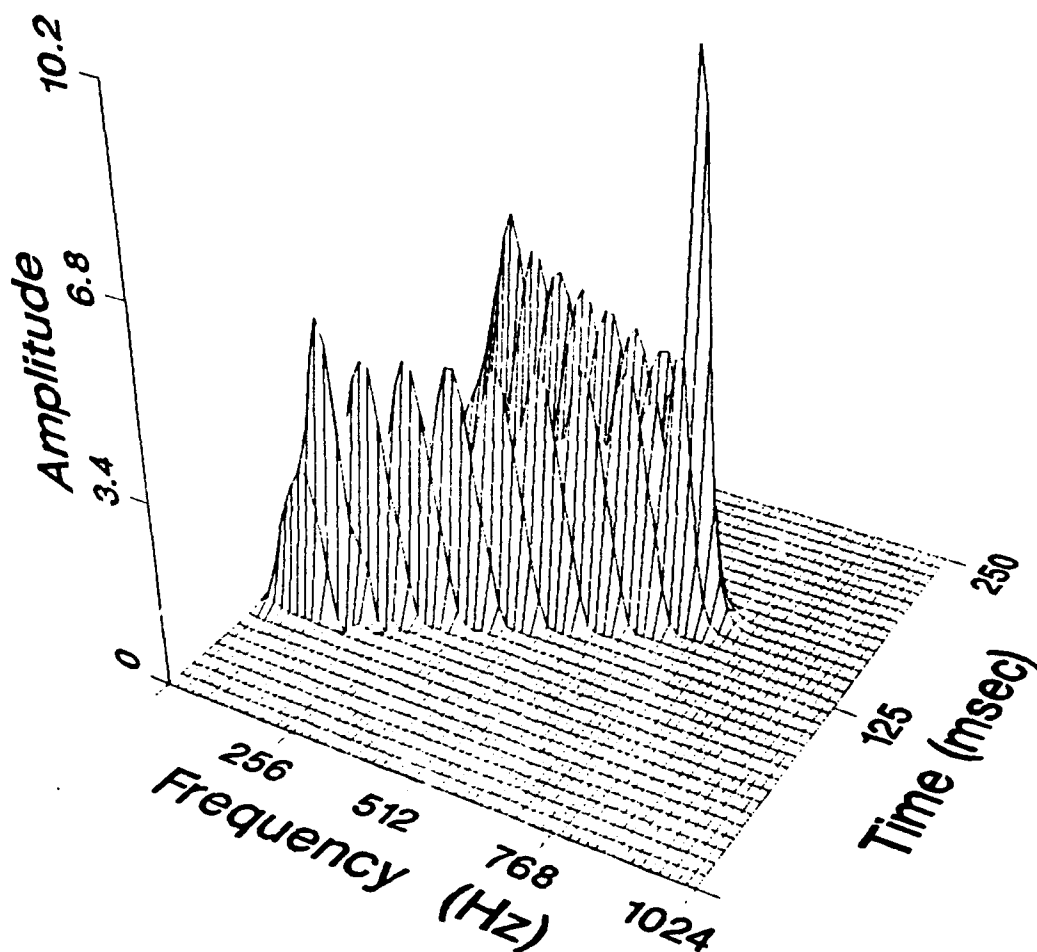
F. WIGFUN3

WIGFUN3 is a postprocessing program with three functions. It reads in the WDF, reduces the output to a desired size, and applies a sliding averaging window to the time-frequency plane.

1. Effect of Reducing the Wigner-Ville Distribution

If the Wigner-Ville Distribution is not reduced and one started with 512 data points, the resulting distributions would be 1024 by 512 points. Graphically this is just too many points to visually deal with. The finest resolution which is understandable is 256 by 128 points. The computer program allows for the following three size reductions; 64 by 32, 128 by 64, and 256 by 128 points. (See Figure 23 on page 39, Figure 24 on page 40, and Figure 25 on page 41.)

Pseudo Wigner-Ville Distribution Linear Chirp

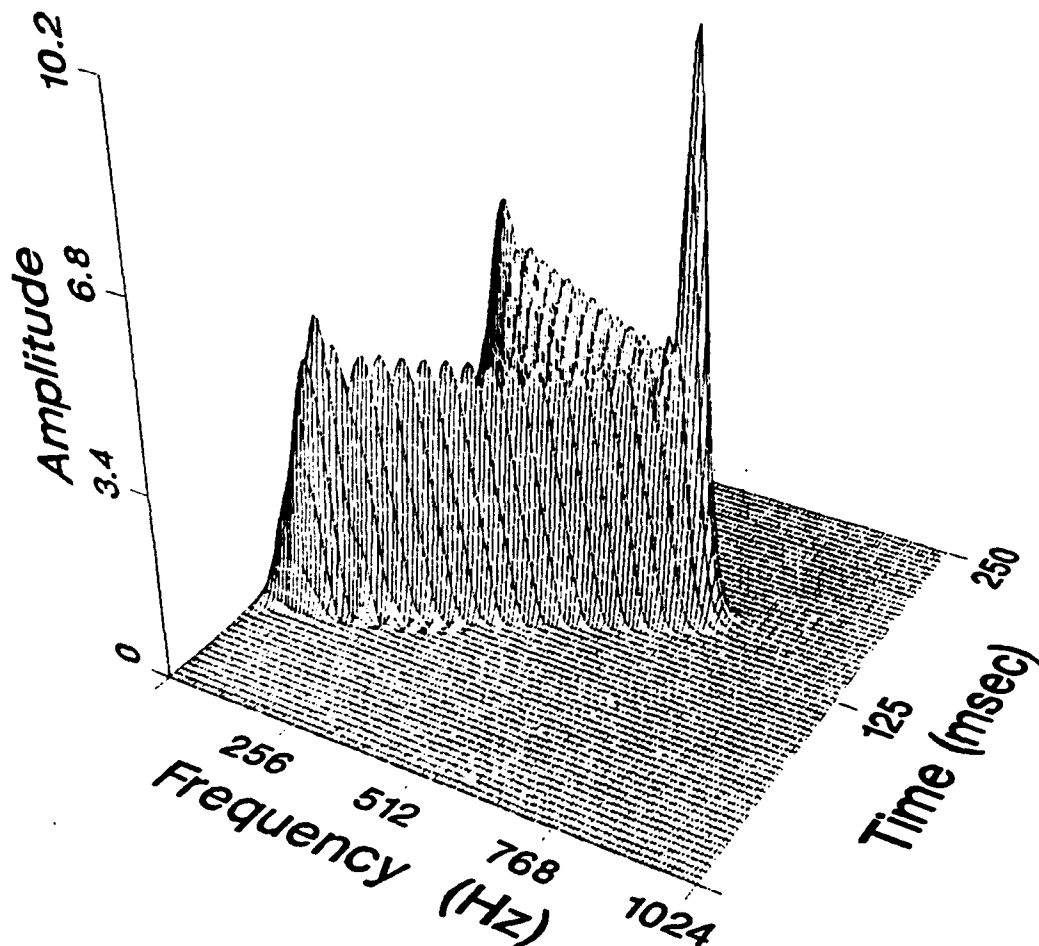


16-APR-1990 09:50:29.07
512 data points
Reduced to 64 x 32
Smoothed 10 x 10

Mean value removed
Time amplified by 0.0
Hamming window time

Figure 23. Pseudo Wigner-Ville Distribution Reduced to 64 x 32 Points

Pseudo Wigner-Ville Distribution Linear Chirp

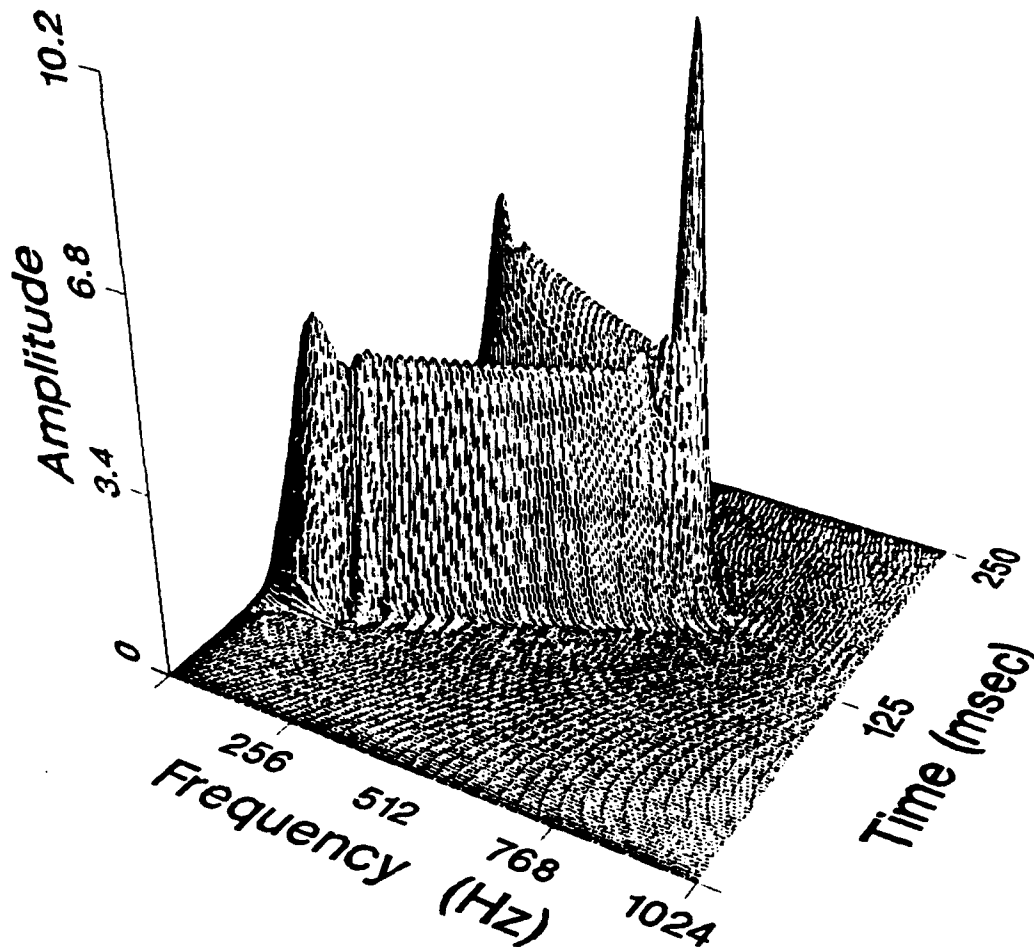


16-APR-1990 11:16:55.56
512 data points
Reduced to 128 x 64
Smoothed 10 x 10

Mean value removed
Time amplified by 0.0
Hamming window time

Figure 24. Pseudo Wigner-Ville Distribution Reduced to 128 x 64 Points

Pseudo Wigner-Ville Distribution Linear Chirp



16-APR-1990 12:17:40.91
512 data points
Reduced to 256 x 128
Smoothed 10 x 10

Mean value removed
Time amplified by 0.0
Hamming window time

Figure 25. Pseudo Wigner-Ville Distribution Reduced to 256 x 128 Points

2. Effect of Smoothing the Wigner-Ville Distribution

As discussed in chapter 2, smoothing the Wigner-Ville Distribution deemphasizes the terms which arise from mathematical calculations and emphasizes the characteristic events. The smoothing, or averaging, process also makes the distribution much more understandable (see Figure 26 on page 43 and Figure 27 on page 44). It is important to note that the smoothing process uses all of the data points which were calculated for the Wigner-Ville Distribution but the size reduction routine only plots the size desired. This averaging also effects the energy representation. Figure 28 on page 45 shows the different views and Figure 29 on page 46 shows the detailed contours of the reduced, but not smoothed, distribution. The sliding averaging window is based on an algorithm used by Wahl and Bolton [Ref. 8]. At a given point on the time-frequency plane the weighted window, hg , is expressed as:

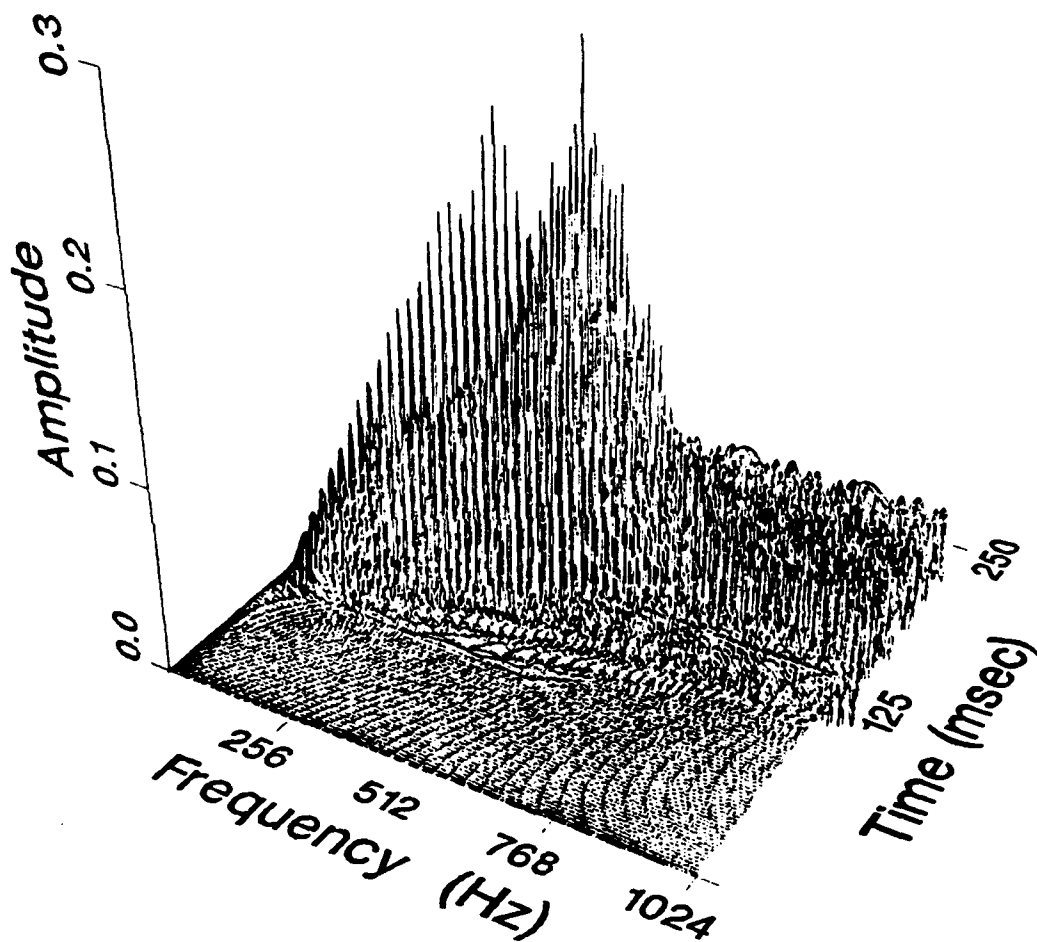
$$hg(i, j) = \frac{1}{400\pi^2 \Delta\omega \Delta t} e^{-\frac{i^2}{200} - \frac{j^2}{200}} \quad (23)$$

where:

- $10 \leq i \leq 10$
- $10 \leq j \leq 10$

The weighted window is applied at each of the reduced data points which are being plotted.

Reduced Wigner-Ville Distribution Linear Chirp

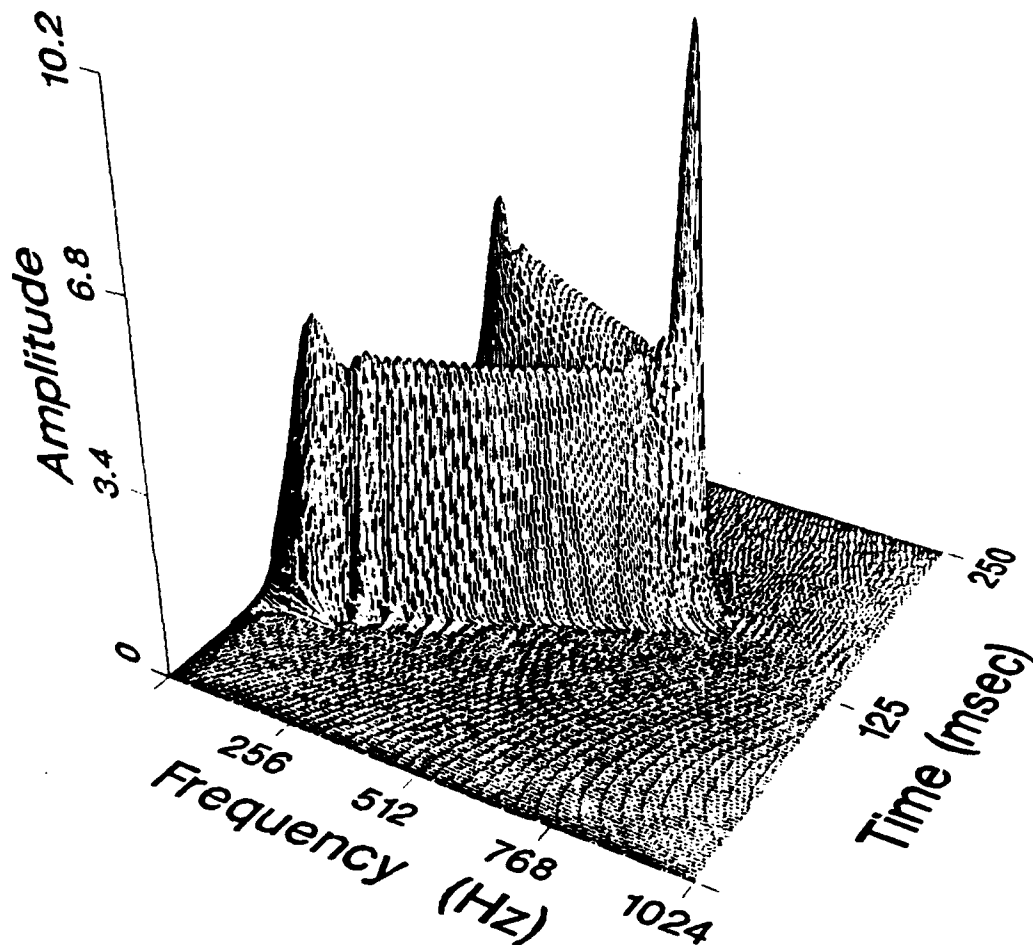


16-APR-1990 12:38:27.17
512 data points
Reduced to 256 x 128

Mean value removed
Time amplified by 0.0
Hamming window time

Figure 26. Linear Chirp with No Smoothing

Pseudo Wigner-Ville Distribution Linear Chirp



16-APR-1990 12:17:40.91
512 data points
Reduced to 256 x 128
Smoothed 10 x 10

Mean value removed
Time amplified by 0.0
Hamming window time

Figure 27. Linear Chirp with Smoothing

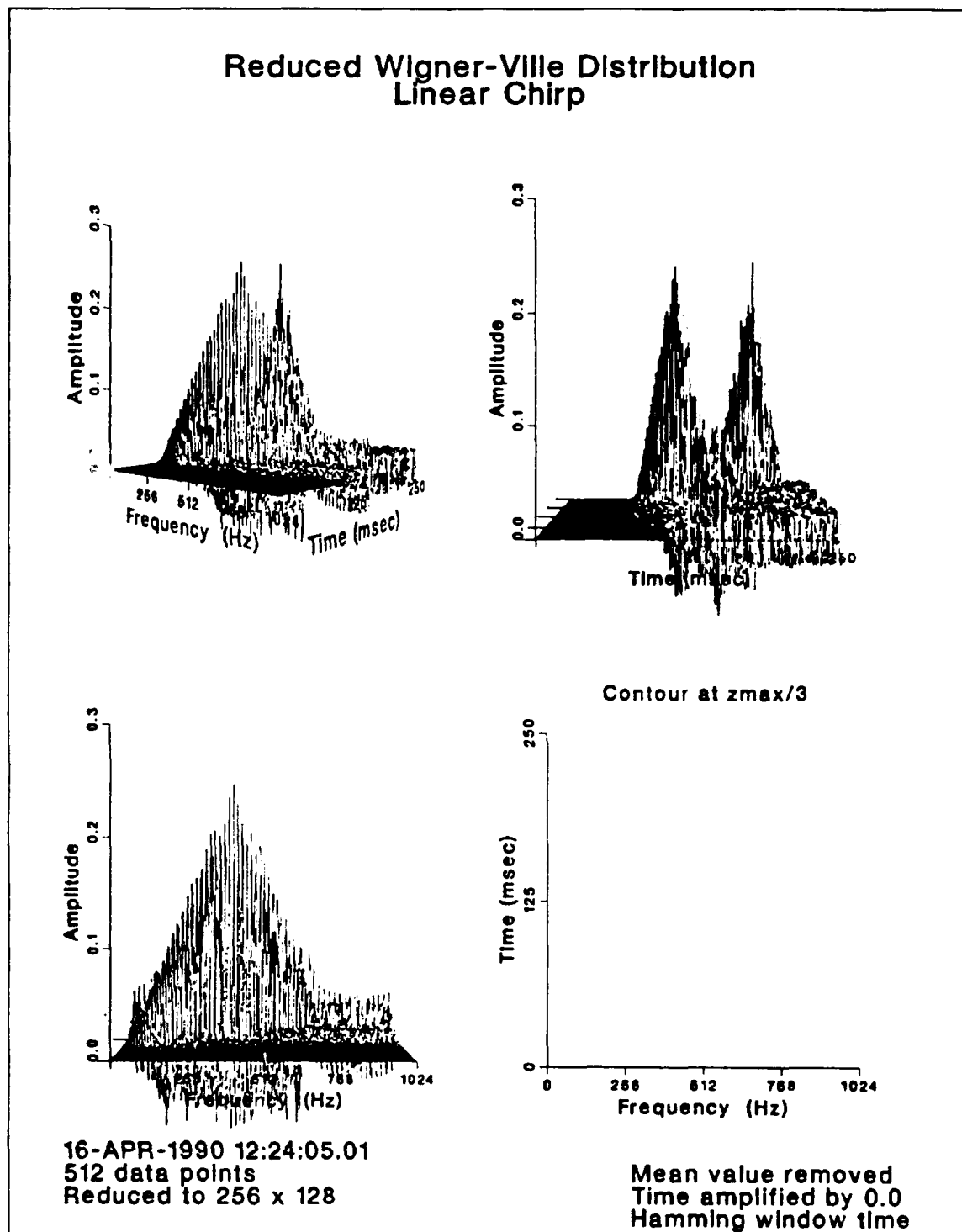


Figure 28. Different Views of Linear Chirp with No Smoothing

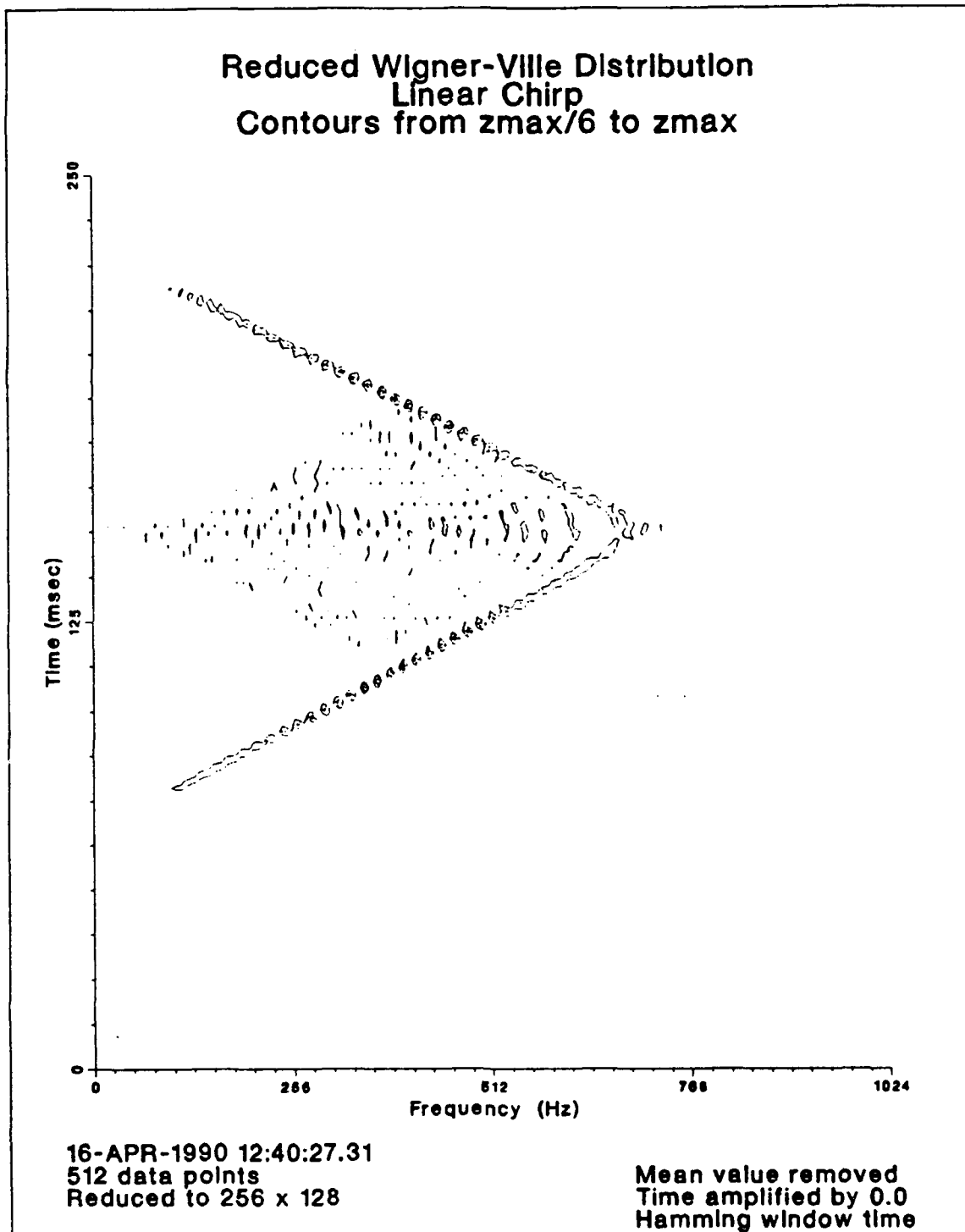


Figure 29. Detailed Contours of Linear Chirp with No Smoothing

G. WIGFUN4A, WIGFUN4B, AND WIGFUN4C

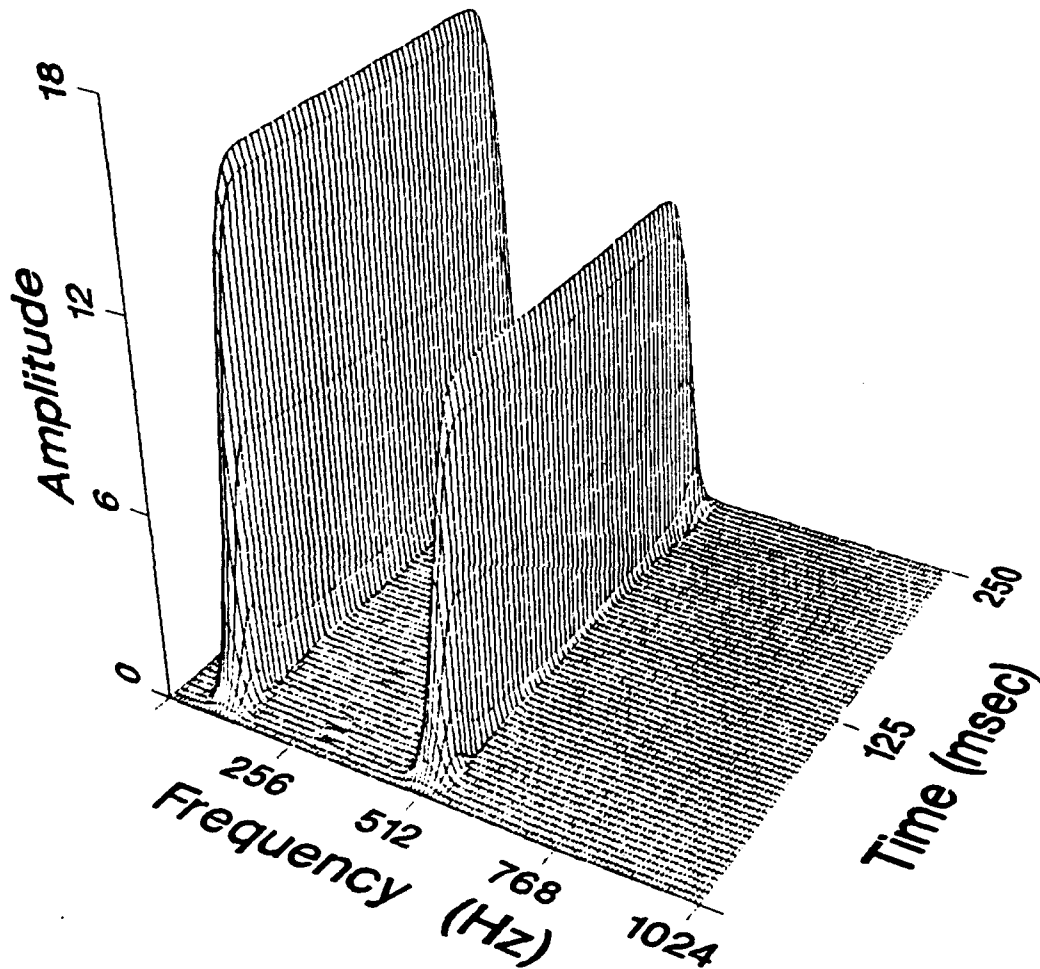
The WIGFUN4 programs plot the resulting distributions. Note that the time axis is changed from seconds to milliseconds. WIGFUN4a plots the 64 by 32 point distributions, WIGFUN4b plots the 128 by 64 point distributions, and WIGFUN4c plots the 256 by 128 point distributions.

H. OTHER CONSIDERATIONS

1. Effect of Noise

Figure 12 on page 23 and Figure 13 on page 24 bring up an important topic, which is, what is the effect of noise on the Pseudo Wigner-Ville Distribution? Figure 30 on page 48 is the Pseudo Wigner-Ville Distribution of two sine waves, 100 Hz and 500 Hz, added together with minimal noise present. Figure 31 on page 49 is the input time signal. Figure 32 on page 50 is the Pseudo Wigner-Ville Distribution of the same two sine waves in a little noise, Figure 33 on page 51 is the input time signal. Figure 34 on page 52 is the Pseudo Wigner-Ville Distribution of the same two sine waves in more noise, Figure 35 on page 53 is the input time signal. By comparing these distributions it can be noted that as the level of noise is increased the amplitudes of the distributions vary but the frequency content remains consistent. Each of the input time signal figures are made up of an upper plot from WIGFUN1 (512 data points) and a lower plot from a Dynamic Signal Analyzer (1024 data points) recorded at digitization.

Pseudo Wigner-Ville Distribution 100 Hz and 500 Hz sin waves



11-APR-1990 13:45:01.97
512 data points
Reduced to 128 x 64
Smoothed 10 x 10

Mean value removed
Time amplified by 0.0
Hamming window time

Figure 30. Pseudo Wigner-Ville Distribution with Minimal Noise

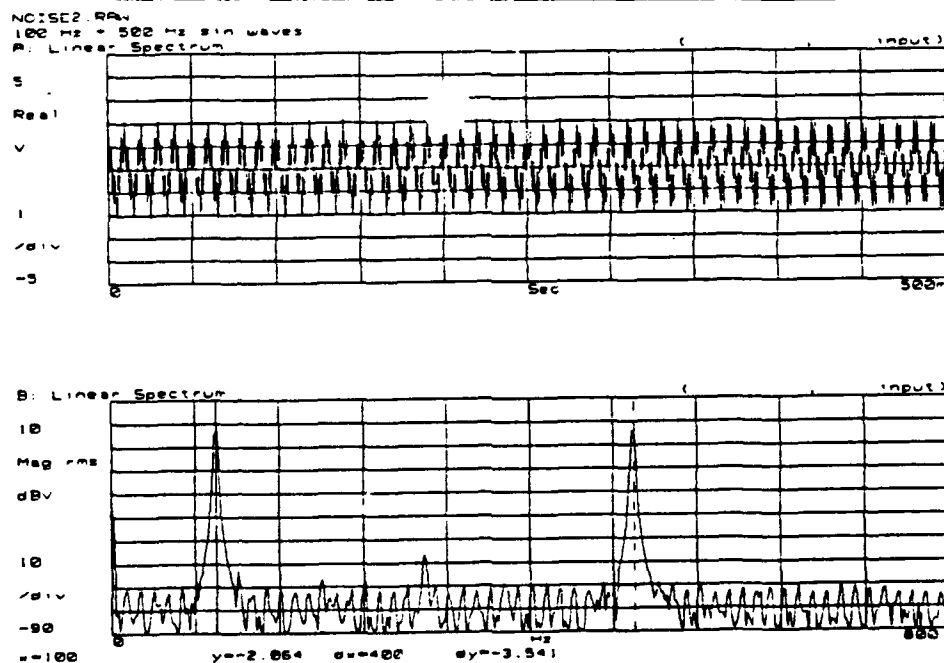
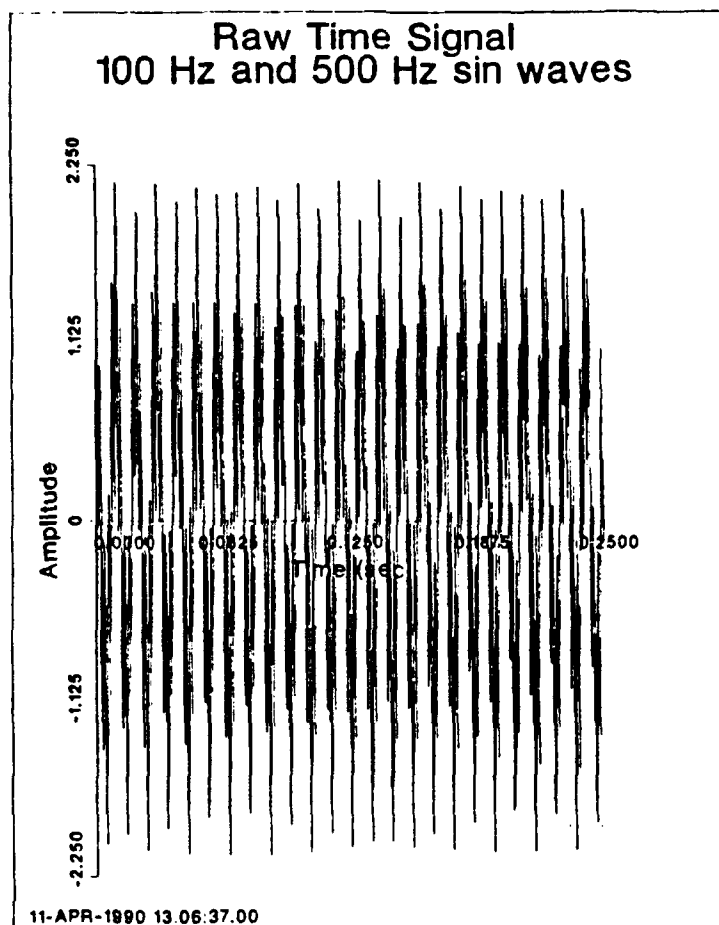
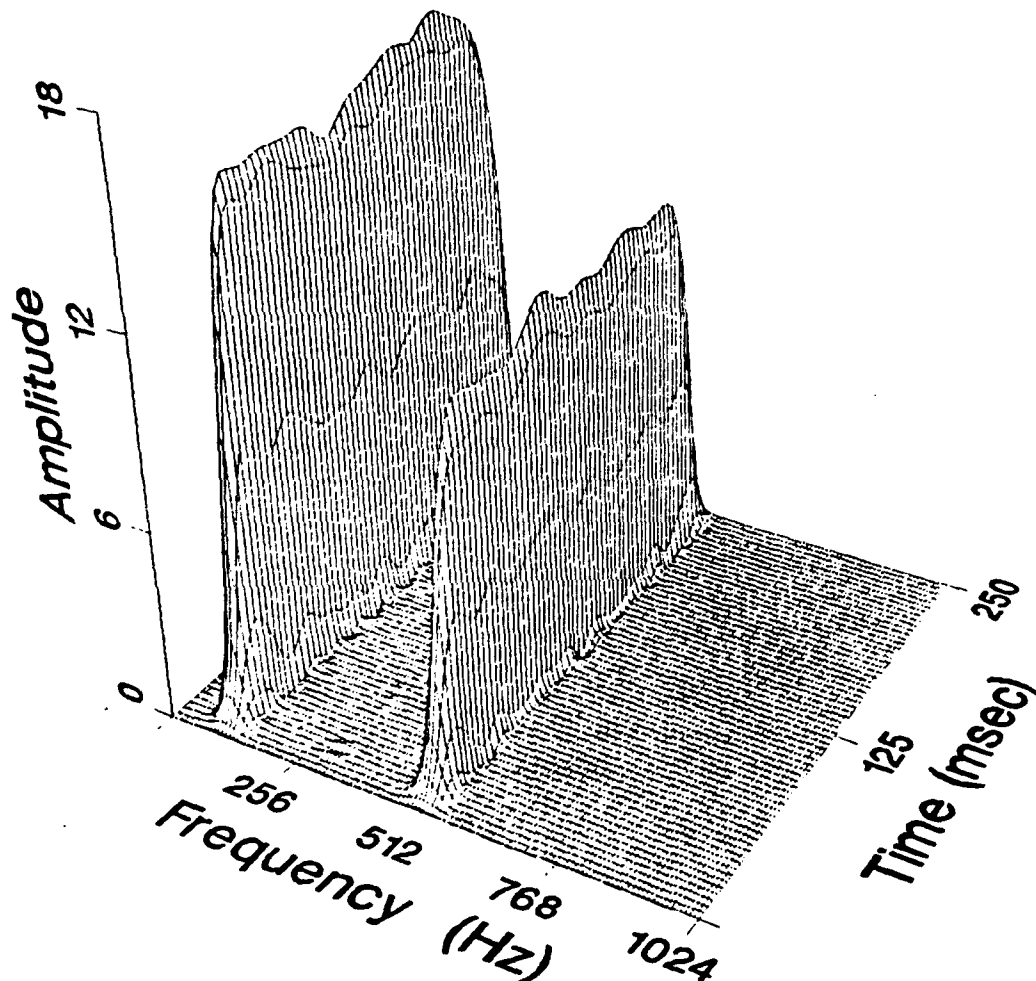


Figure 31. Input Time Signal with Minimal Noise

Pseudo Wigner-Ville Distribution 100 Hz and 500 Hz sin waves in noise

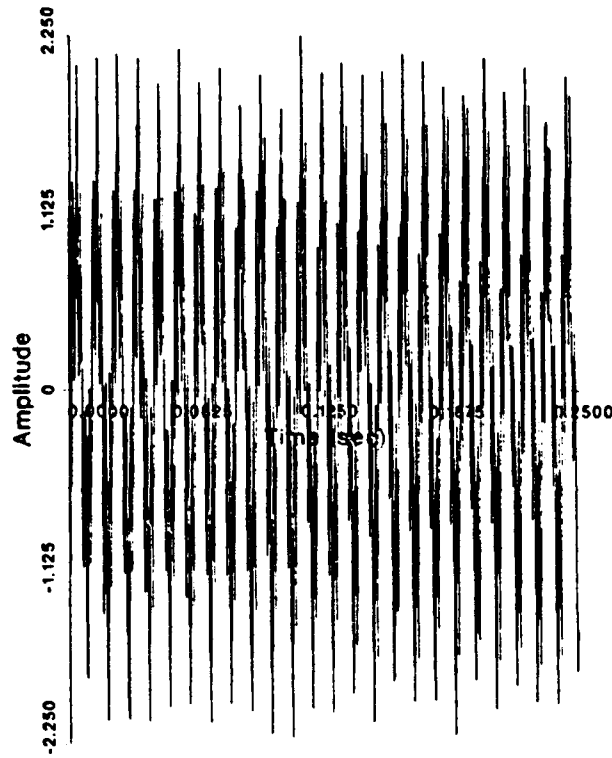


11-APR-1990 14:30:35.97
512 data points
Reduced to 128 x 64
Smoothed 10 x 10

Mean value removed
Time amplified by 0.0
Hamming window time

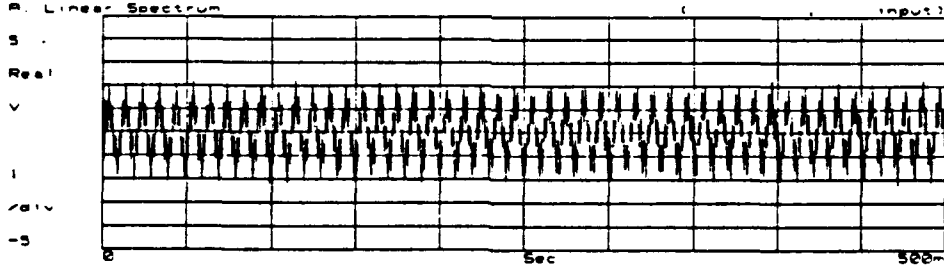
Figure 32. Pseudo Wigner-Ville Distribution with Noise

Raw Time Signal 100 Hz and 500 Hz sin waves in noise



11-APR-1990 13:51:24.25

NOISE3.RAW
100 Hz - 500 Hz sin waves - Noise
A: Linear Spectrum



B: Linear Spectrum

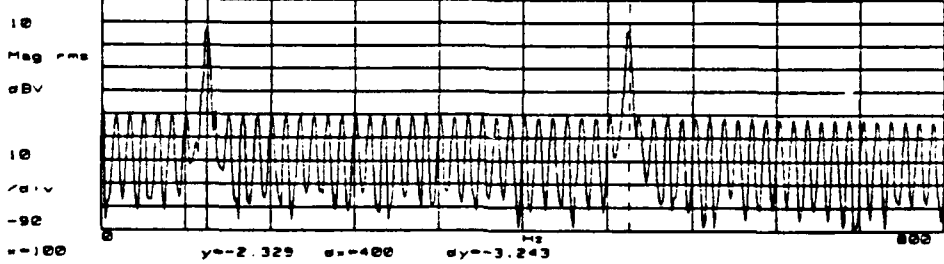
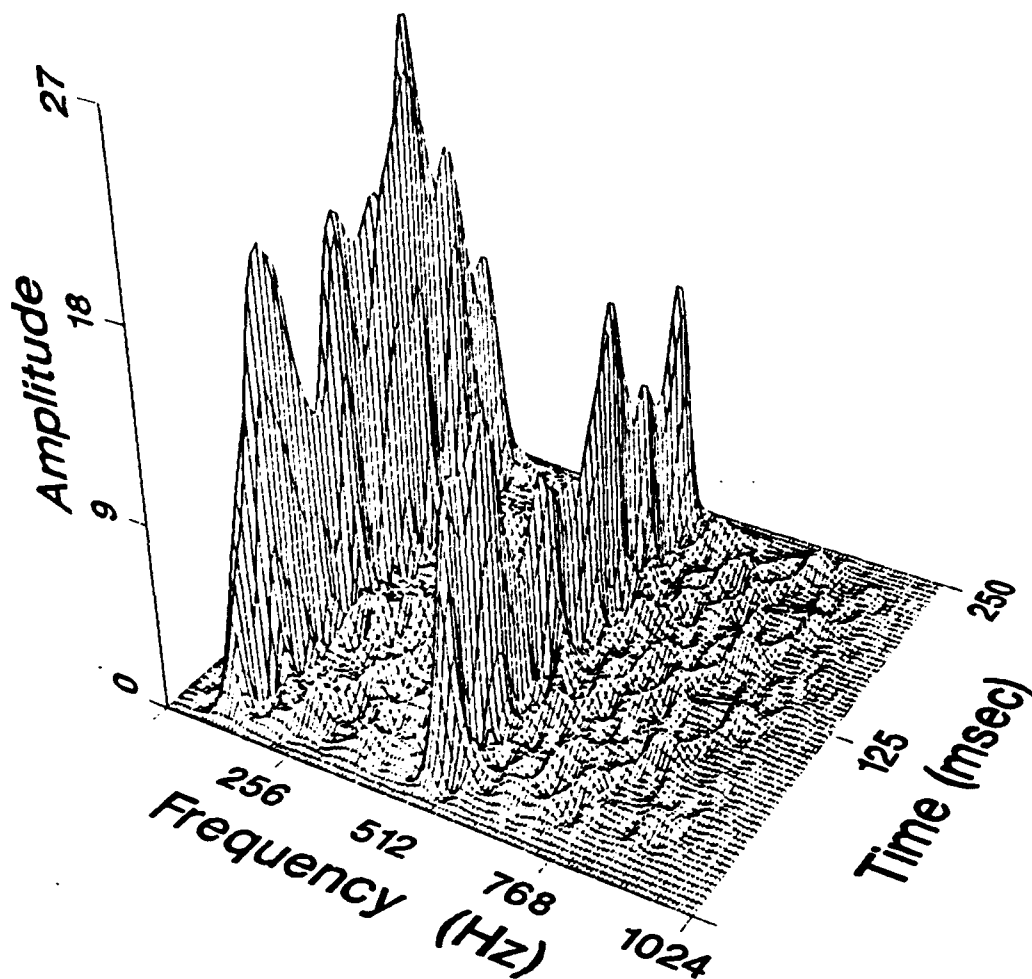


Figure 33. Input Time Signal with Noise

Pseudo Wigner-Ville Distribution 100 Hz and 500 Hz sin waves in noise



11-APR-1990 14:36:00.50
512 data points
Reduced to 128 x 64
Smoothed 10 x 10

Mean value removed
Time amplified by 0.0
Hamming window time

Figure 34. Pseudo Wigner-Ville Distribution with More Noise

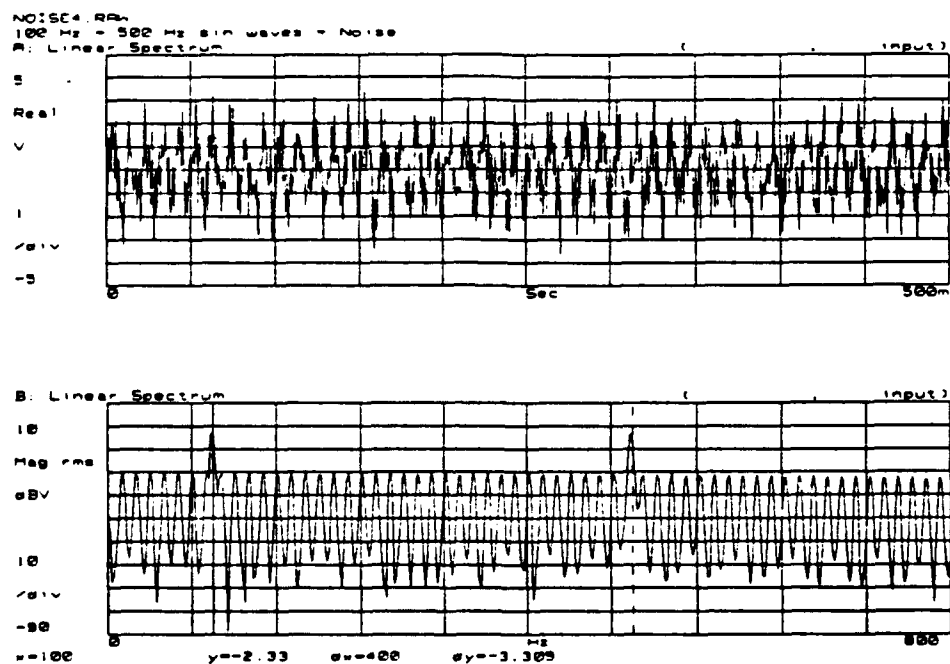
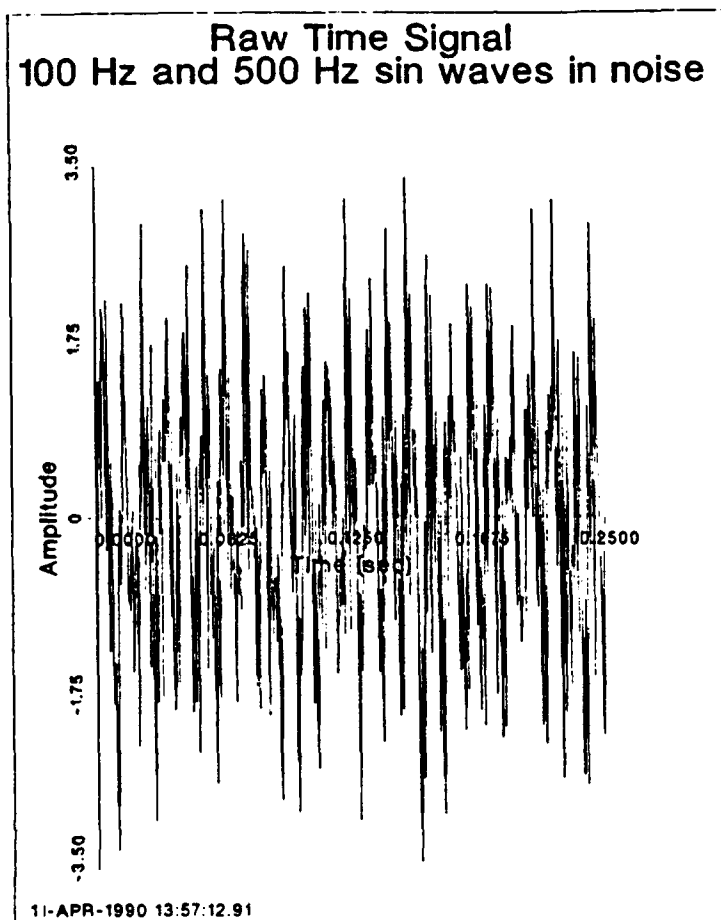
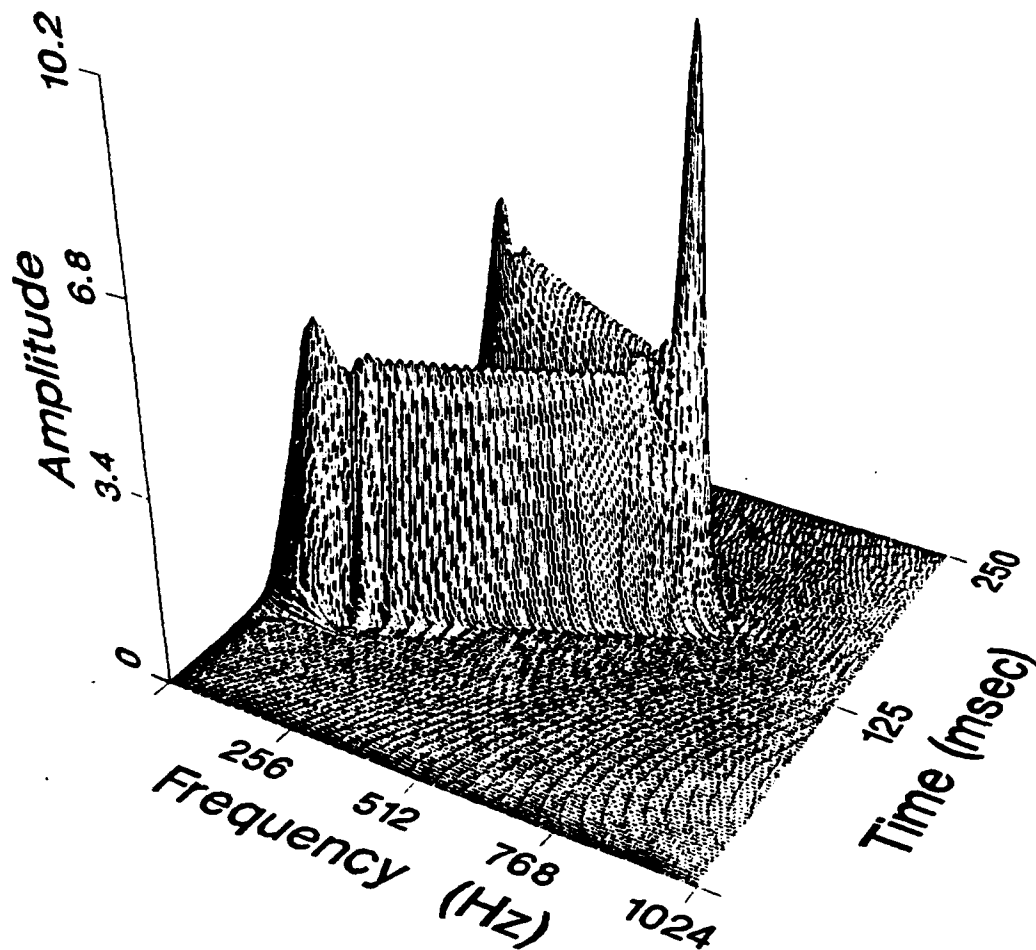


Figure 35. Input Time Signal with More Noise

2. Distribution Interpretation

The interpretation of the Pseudo Wigner-Ville Distribution can be very confusing. Before an error is suspected or if doubt is raised as to the accuracy of the programs, verify the input time domain signal. The Pseudo Wigner-Ville Distribution will accurately portray the input, however, the distribution which results may not be expected. Figure 36 on page 55 is the distribution from the input time signal, Figure 37 on page 56. Figure 38 on page 57 is the distribution from the input time signal, Figure 39 on page 58. From looking at the input time domain signals, the same Pseudo Wigner-Ville Distribution could be expected, however, they are not. The peaks in Figure 38 result from the extra data before and after the chirp in the time signal.

Pseudo Wigner-Ville Distribution Linear Chirp



16-APR-1990 12:17:40.91
512 data points
Reduced to 256 x 128
Smoothed 10 x 10

Mean value removed
Time amplified by 0.0
Hamming window time

Figure 36. Pseudo Wigner-Ville Distribution

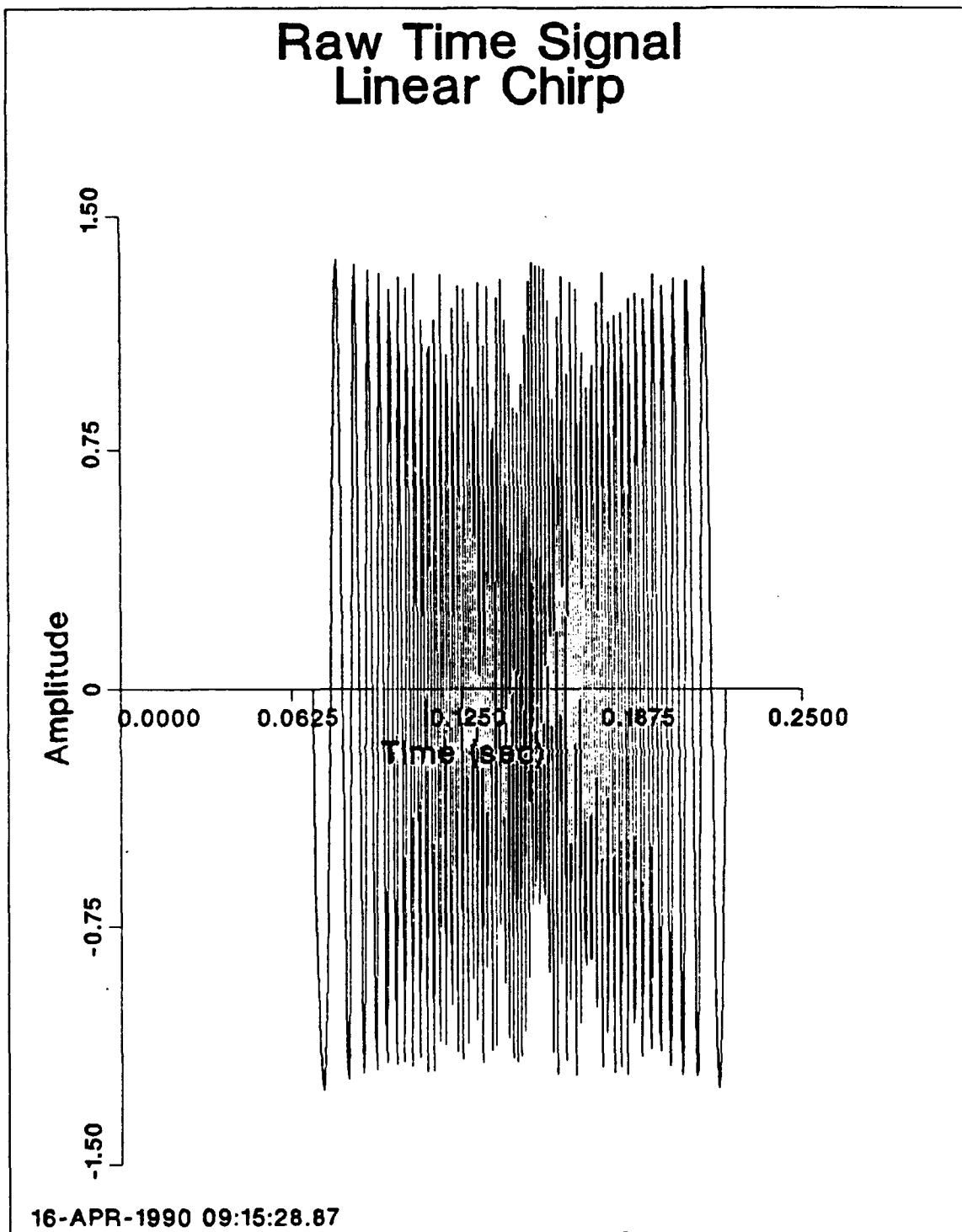
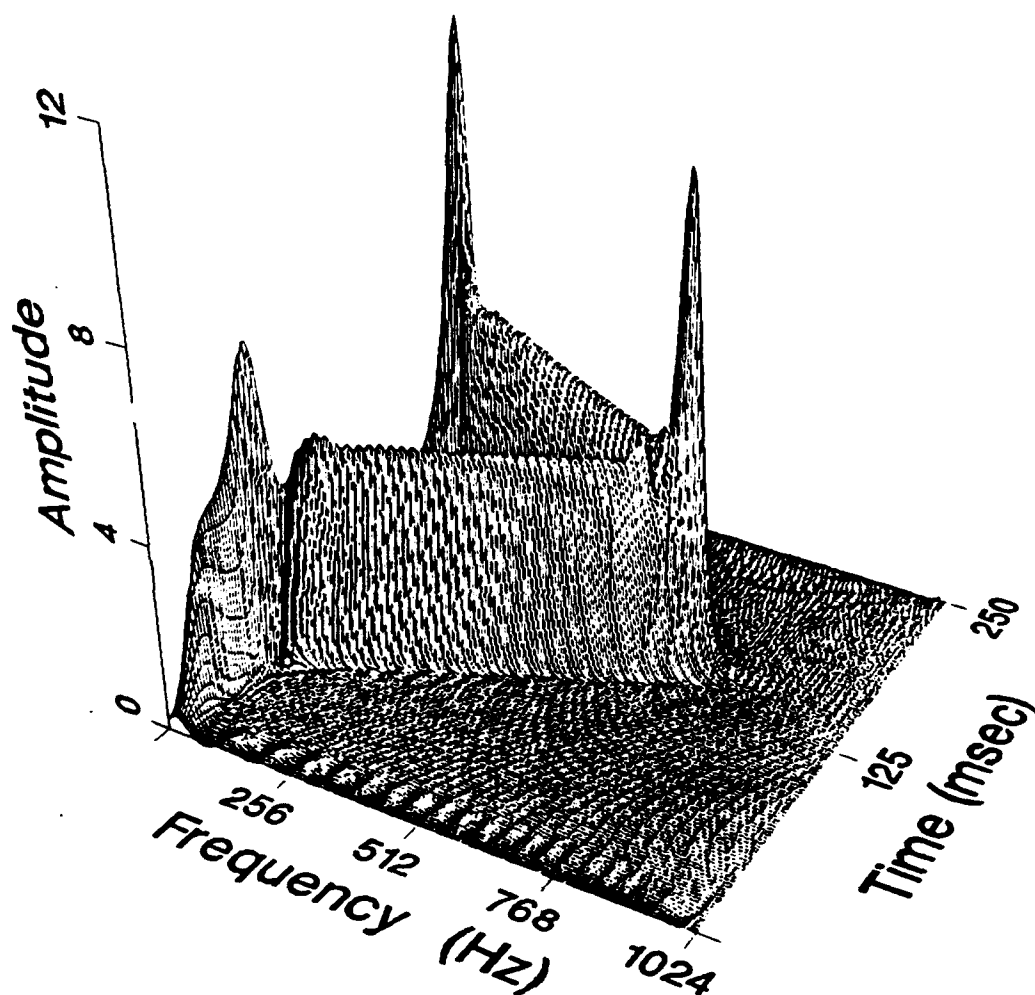


Figure 37. Input Time Signal

Pseudo Wigner-Ville Distribution Linear Chirp - A



2-APR-1990 16:25:38.73
512 data points
Reduced to 256 x 128
Smoothed 10 x 10

Mean value removed
Time amplified by 0.0
Hamming window time

Figure 38. Pseudo Wigner-Ville Distribution

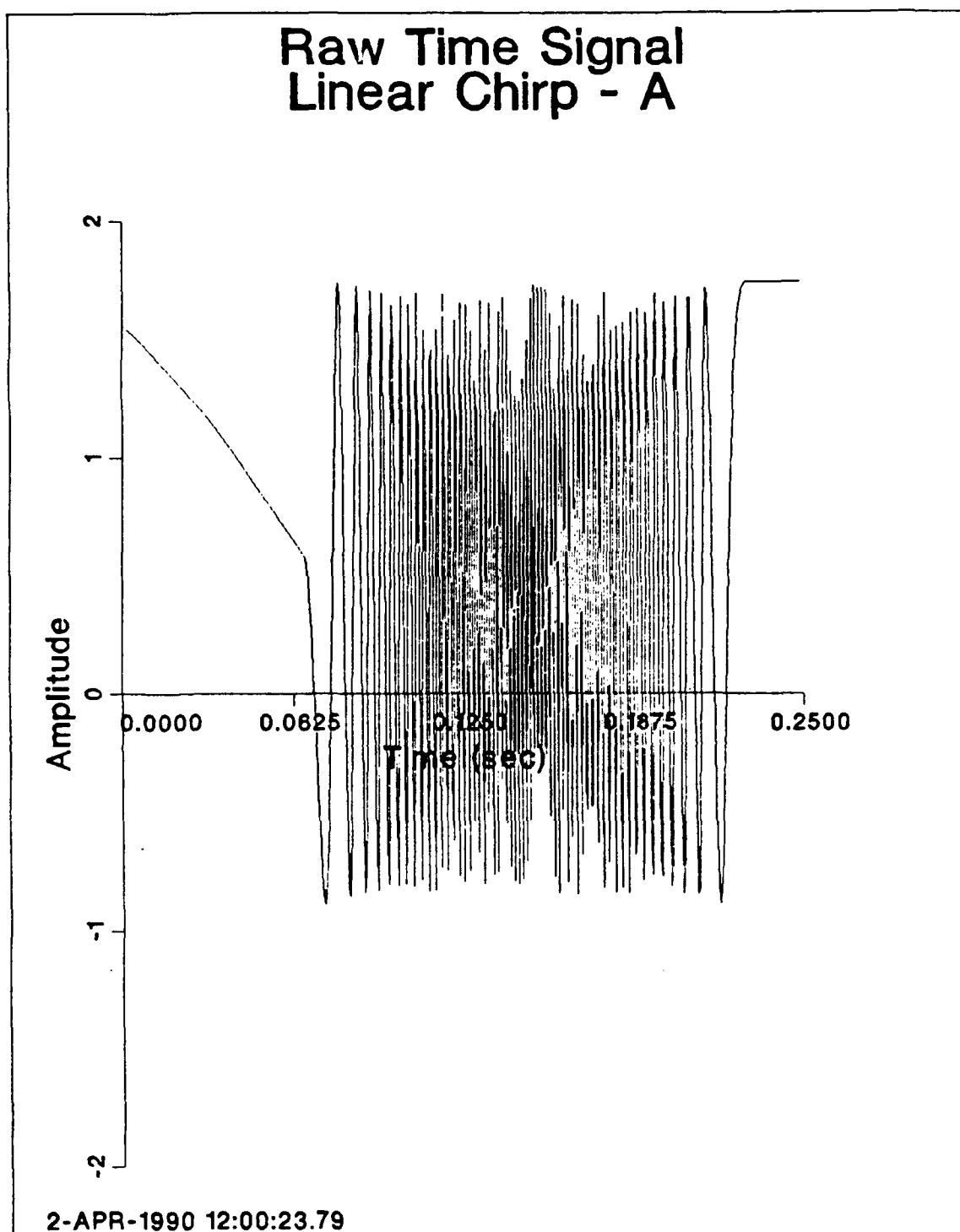
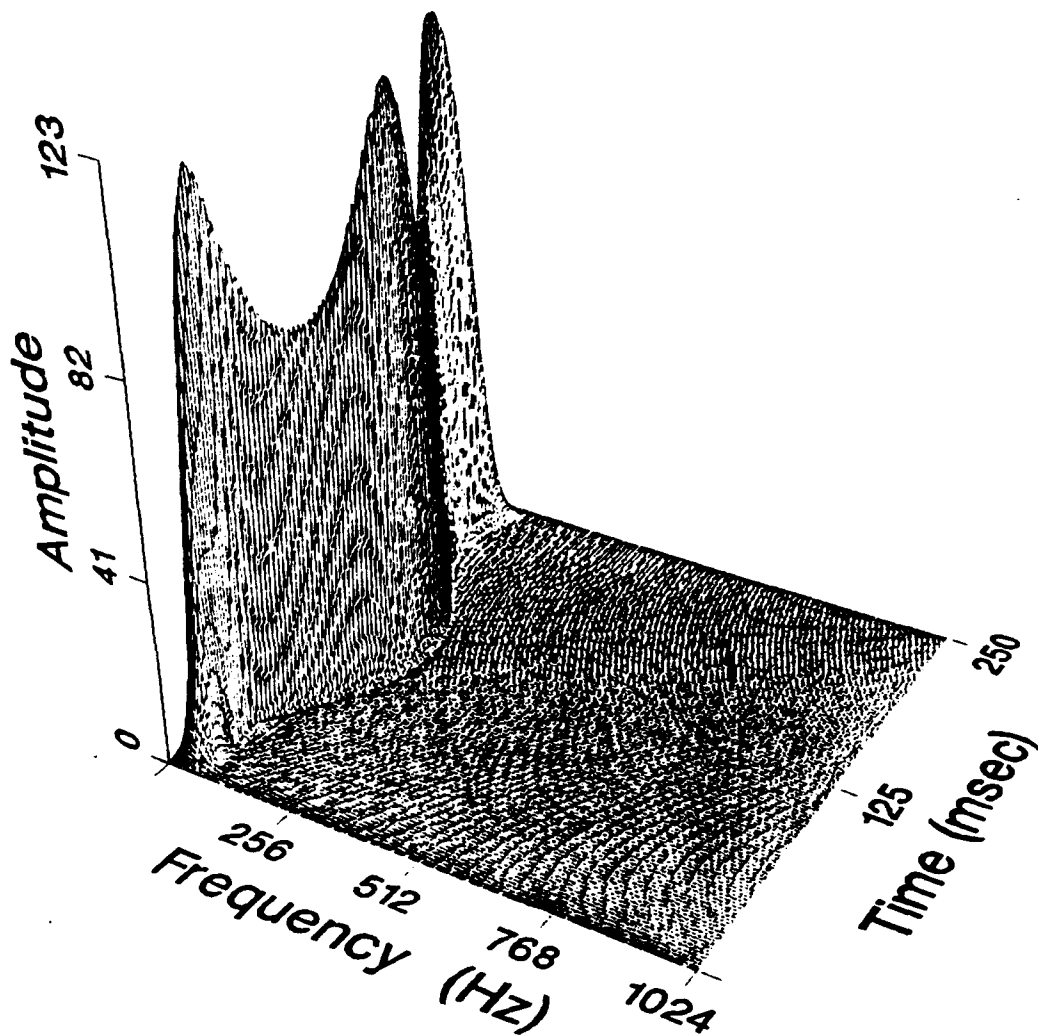


Figure 39. Input Time Signal

3. Swept Sine Wave Example

Figure 40 on page 60 through Figure 45 on page 65 were produced from a sine wave whose frequencies were varied with a sine wave. They demonstrate the effect of varying the Δt and the ability to portray time dependent frequencies. Note the presence of the pronounced peak values located where the sweep of the frequencies are changed from positive to negative and negative to positive.

Pseudo Wigner-Ville Distribution Swept Sine Wave



14-APR-1990 15:12:54.98
512 data points
Reduced to 256 x 128
Smoothed 10 x 10

Mean value removed
Time amplified by 0.0
Hamming window time

Figure 40. Swept Sine Wave, Frequency Span 0-1024 Hz

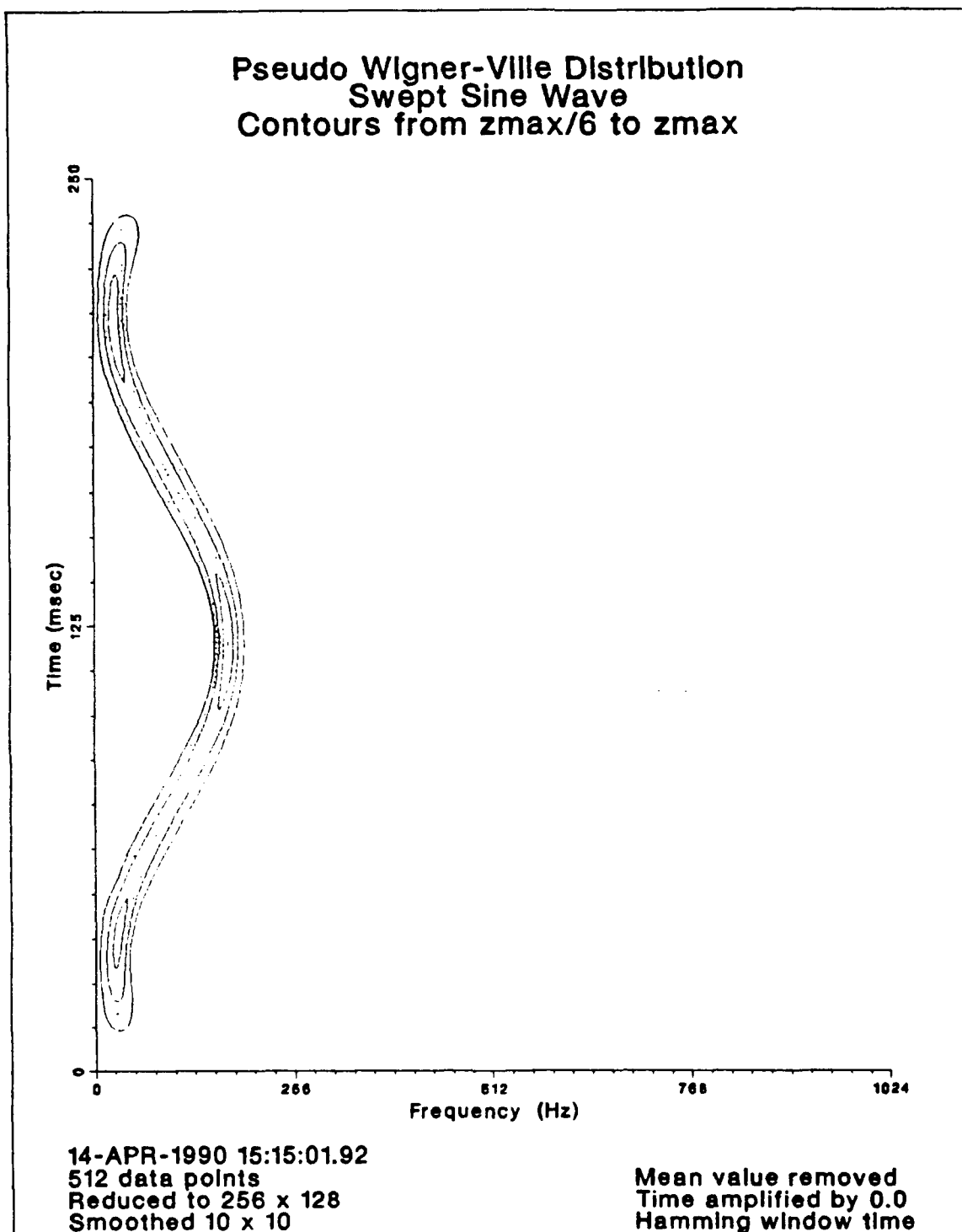


Figure 41. Detailed Contour Plot Swept Sine Wave, Frequency Span 0-1024 Hz

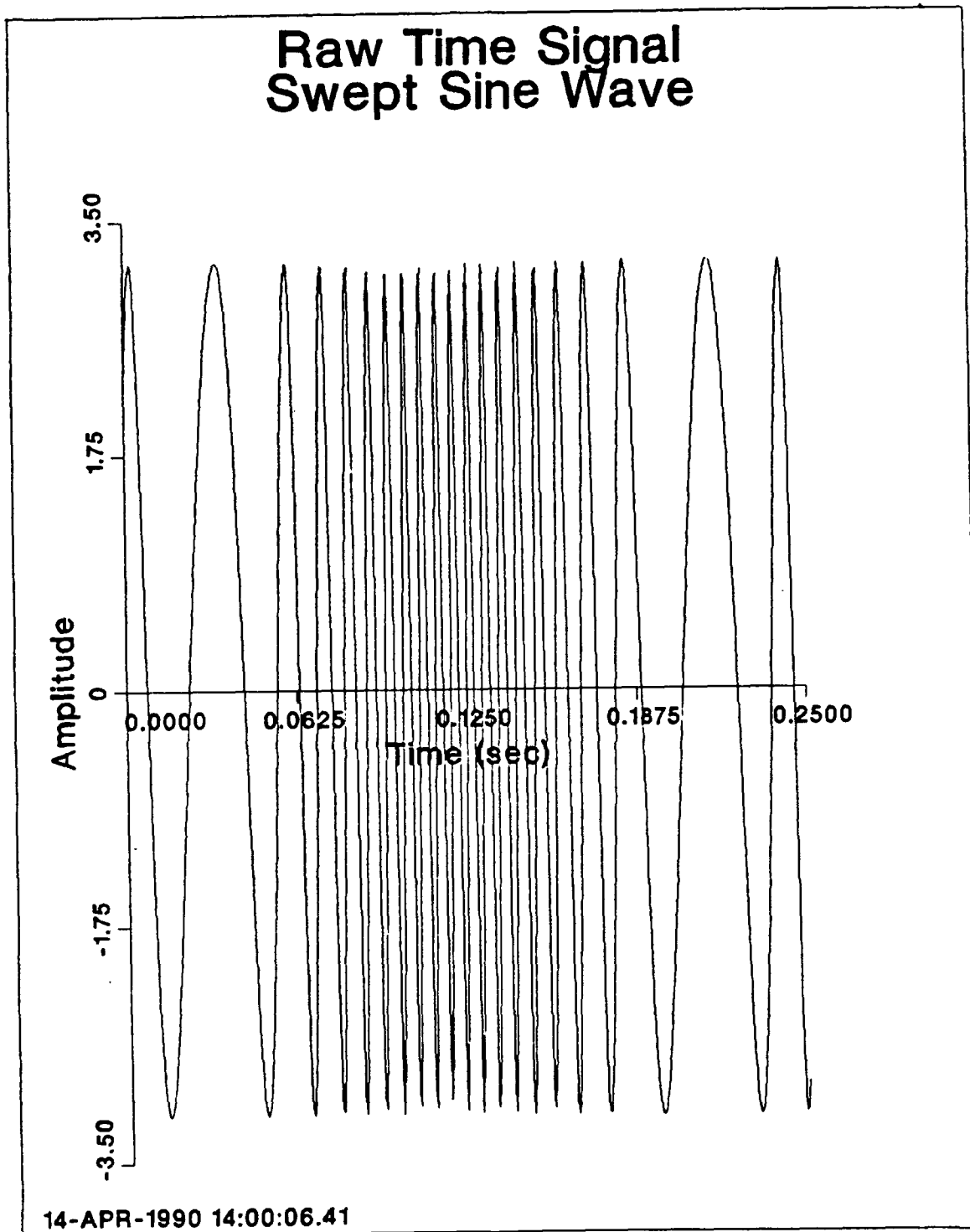
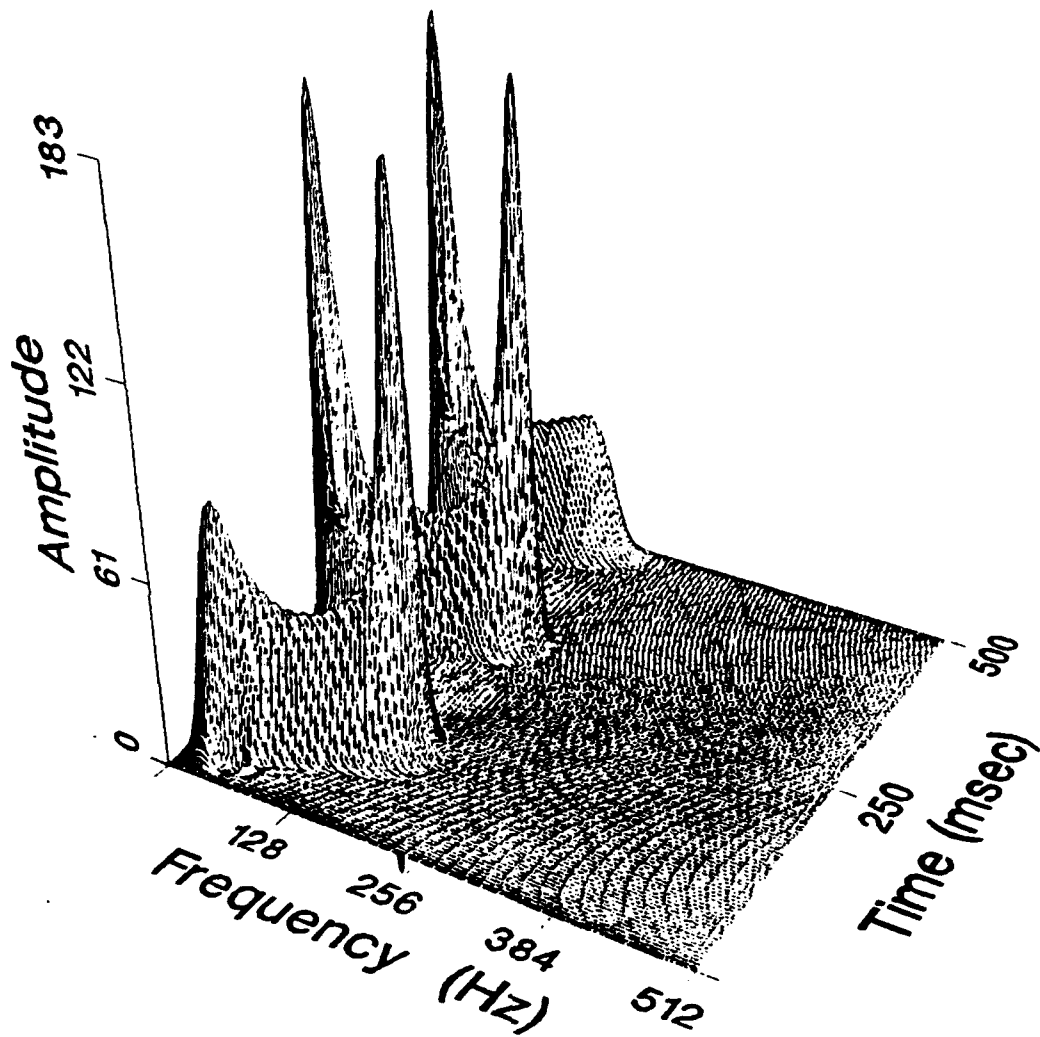


Figure 42. Input Time Signal for Swept Sine Wave, Frequency Span 0-1024 Hz

Pseudo Wigner-Ville Distribution Swept Sine Wave



14-APR-1990 17:28:28.80
512 data points
Reduced to 256 x 128
Smoothed 10 x 10

Mean value removed
Time amplified by 0.0
Hamming window time

Figure 43. Swept Sine Wave, Frequency Span 0-512 Hz

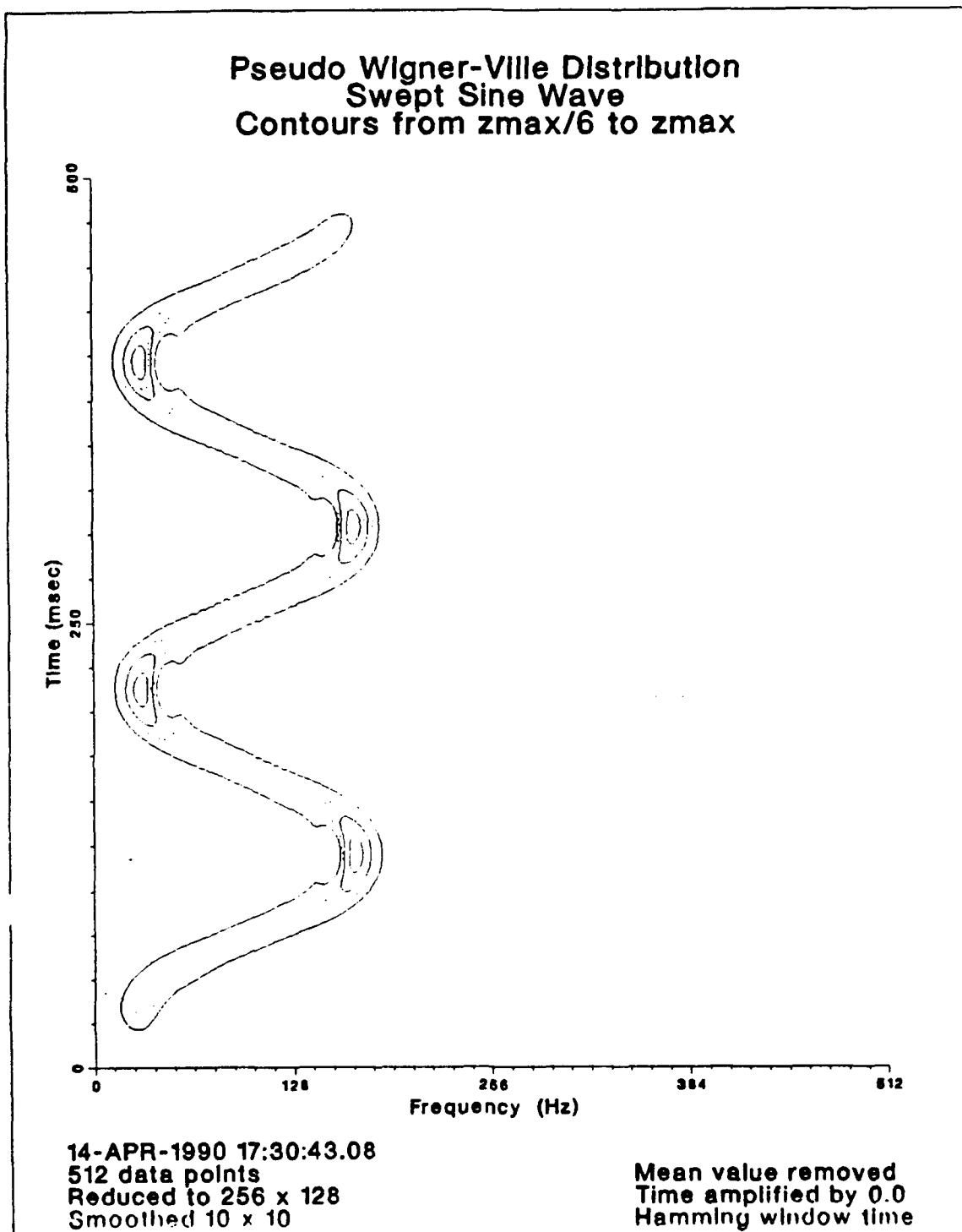


Figure 44. Detailed Contour Plot Swept Sine Wave, Frequency Span 0-512 Hz

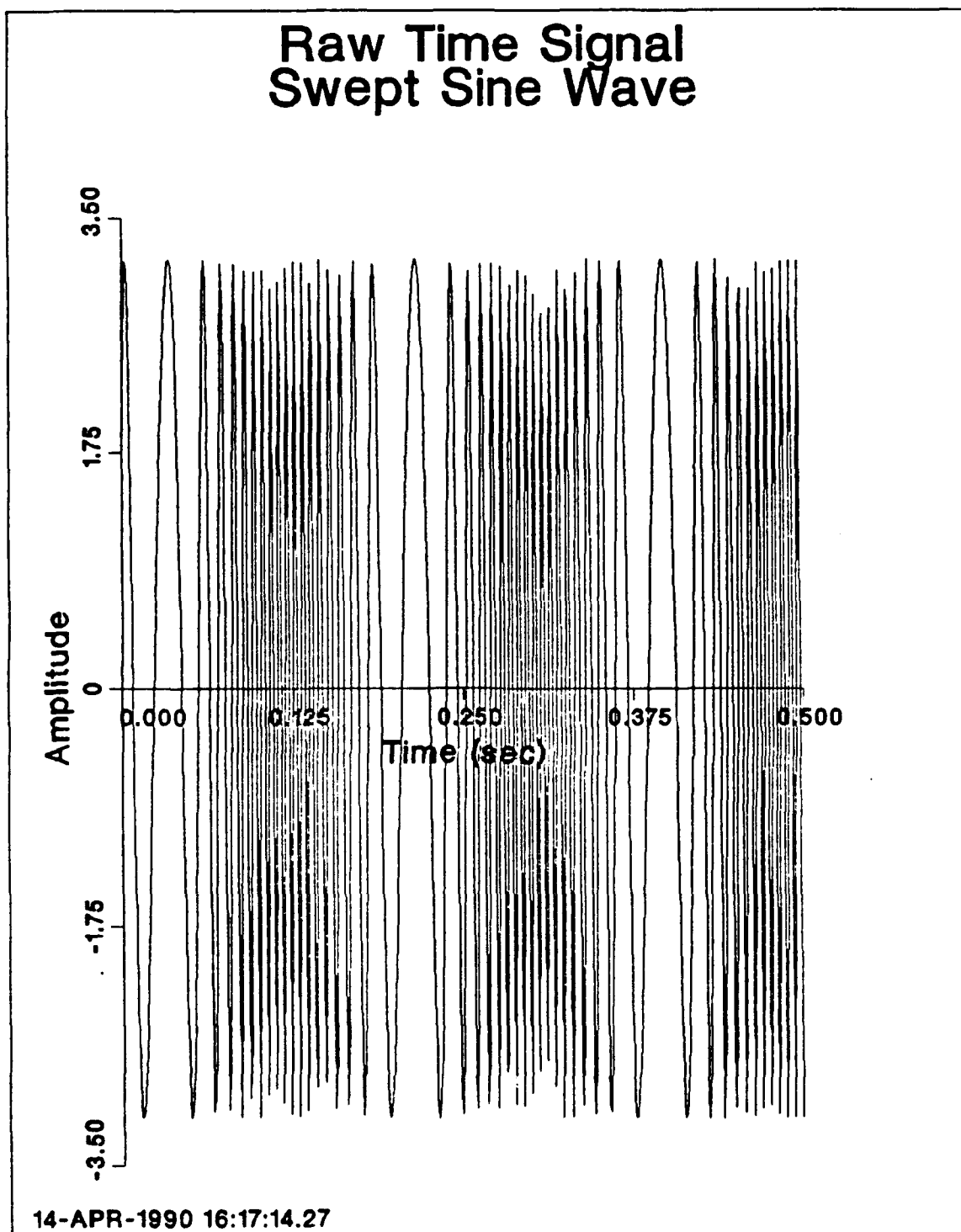


Figure 45. Input Time Signal for Swept Sine Wave, Frequency Span 0-512 Hz

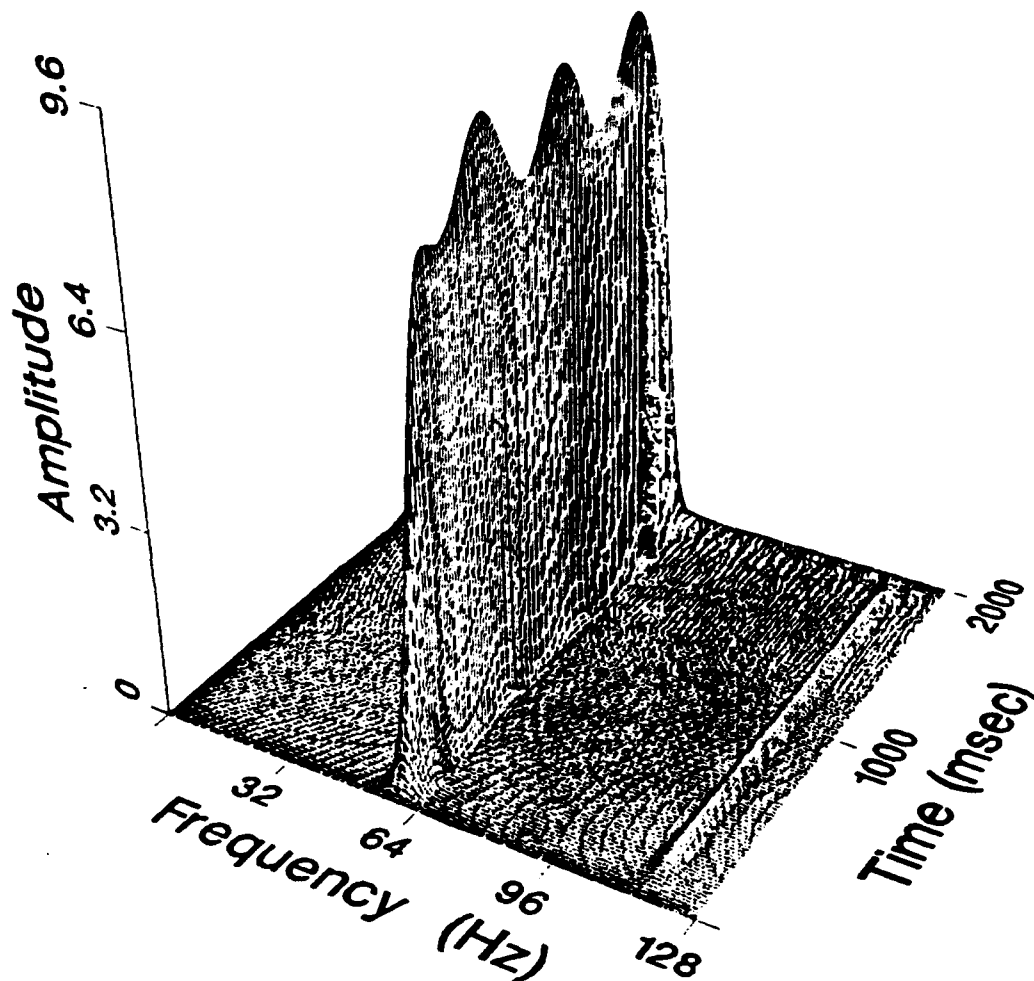
IV. PRACTICAL EXAMPLES USING PSEUDO WIGNER-VILLE DISTRIBUTION

The figures in this chapter represent data taken from an electrically driven single-stage centrifugal pump. The pump speed measurements are a result of data input from a proximity probe measuring the proximity to a gear directly coupled to the shaft.

Figure 46 on page 67 through Figure 51 on page 72 show the pump operating at a steady state condition with flow rates of 10 gallons per minute and 100 gallons per minute. Note that since the flow rate is constant, the speed of the pump is constant for all time. The speed of the pump for high and low flow rates is not very different, but it can be seen in the detailed contour plots that for 10 GPM the speed is about 52 Hz and for 100 GPM the speed is about 48 Hz.

Figure 52 through Figure 63 show the pump in transient operations. Figure 52 on page 73 through Figure 55 on page 76 show the pump speeding up and then steadying out at a constant speed. Figure 56 on page 77 through Figure 59 on page 80 show the pump slowing down from a constant speed. Figure 60 on page 81 through Figure 63 on page 84 show the pump operating at a constant speed, slowing down after being turned off, quickly speeding up when turned back on, and steadying out again.

Pseudo Wigner-Ville Distribution Pump Speed - Steady State 10 GPM



14-APR-1990 18:49:30.16
512 data points
Reduced to 256 x 128
Smoothed 10 x 10

Mean value removed
Time amplified by 5.0
Hamming window time

Figure 46. Pump Speed Steady State 10 GPM

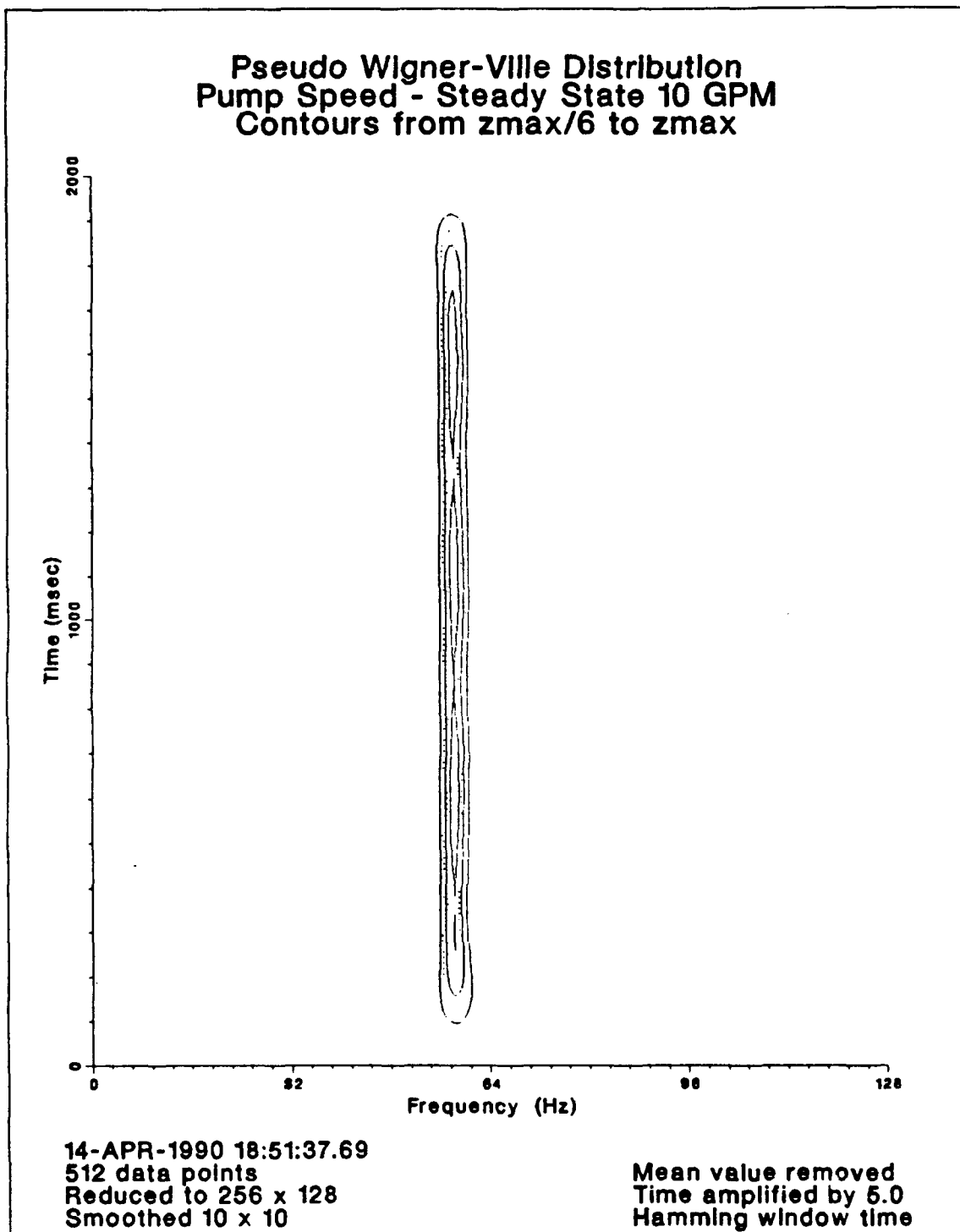
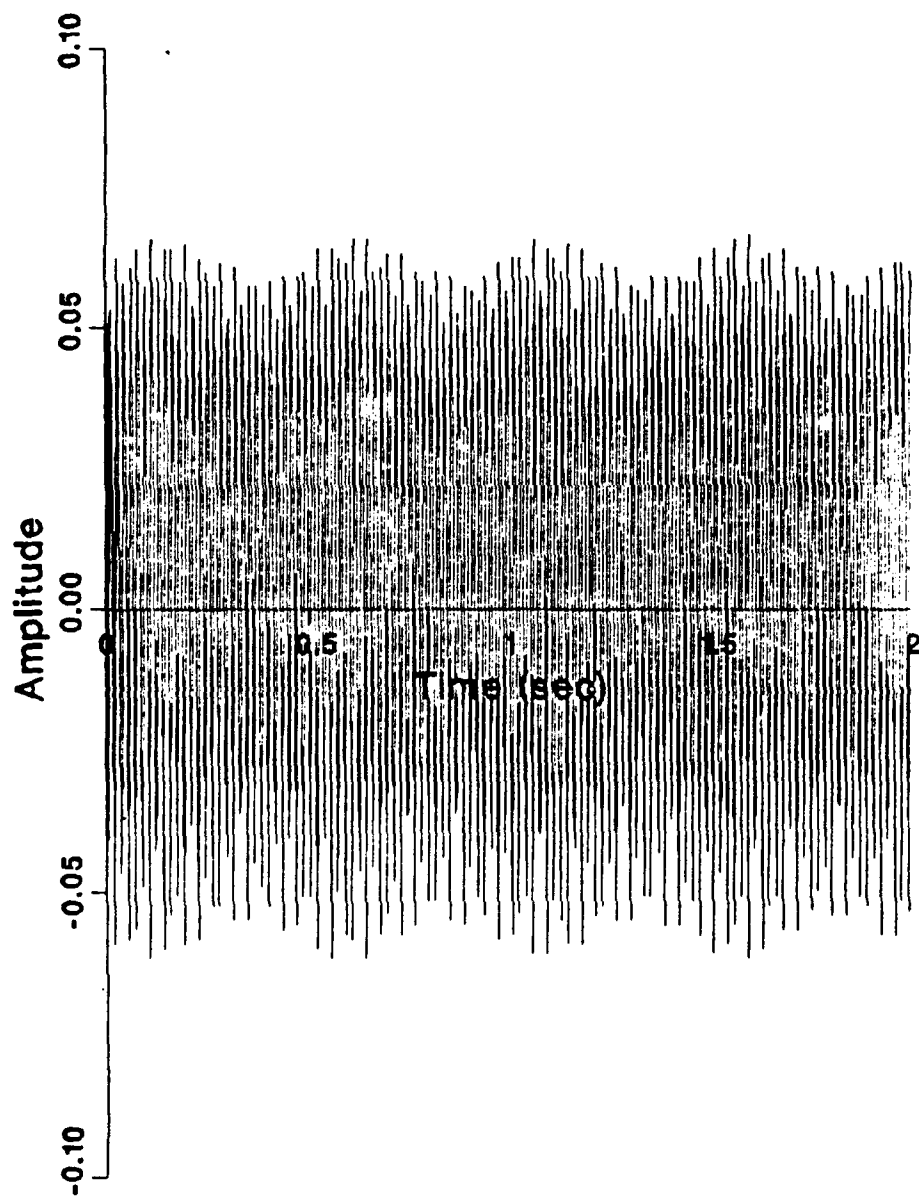


Figure 47. Detailed Contours of Steady State 10 GPM

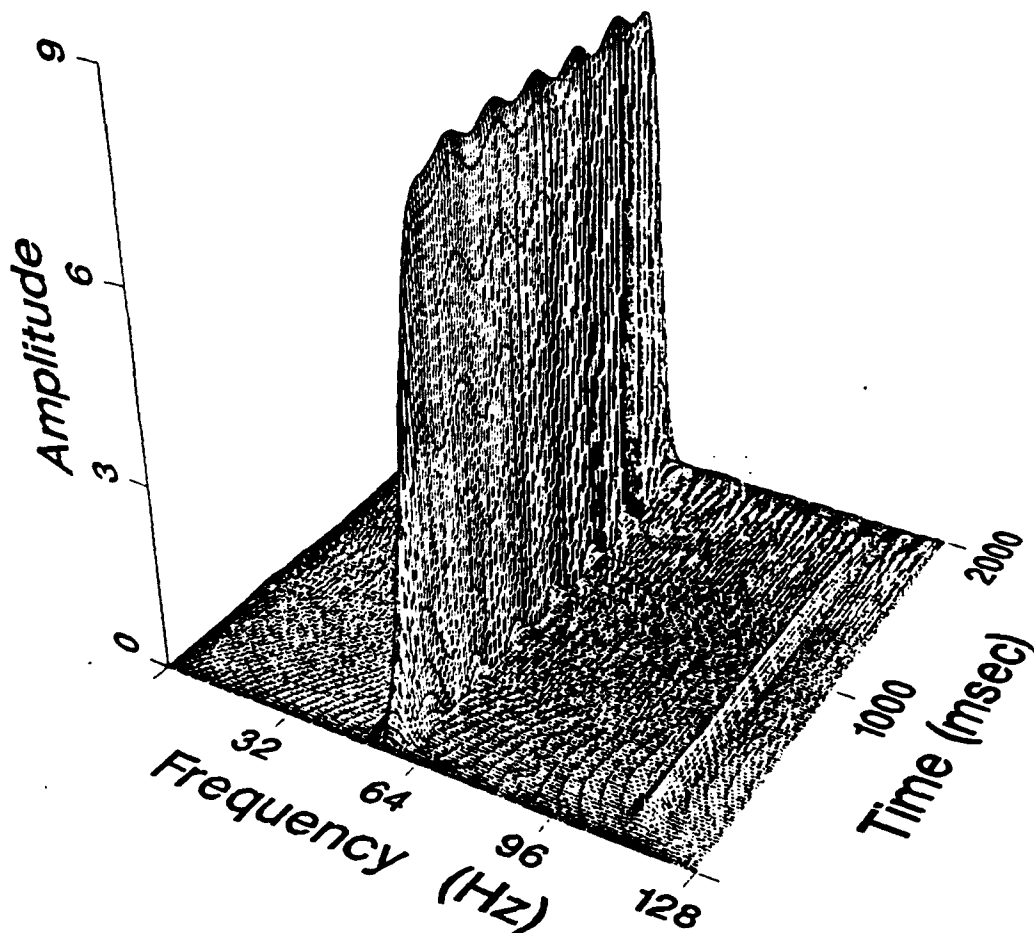
Raw Time Signal Pump Speed - Steady State 10 GPM



14-APR-1990 17:42:05.09

Figure 48. Input Time Signal Steady State 10 GPM

Pseudo Wigner-Ville Distribution Pump - Steady State 100 GPM



14-APR-1990 16:58:07.94
512 data points
Reduced to 256 x 128
Smoothed 10 x 10

Mean value removed
Time amplified by 5.0
Hamming window time

Figure 49. Pump Speed Steady State 100 GPM

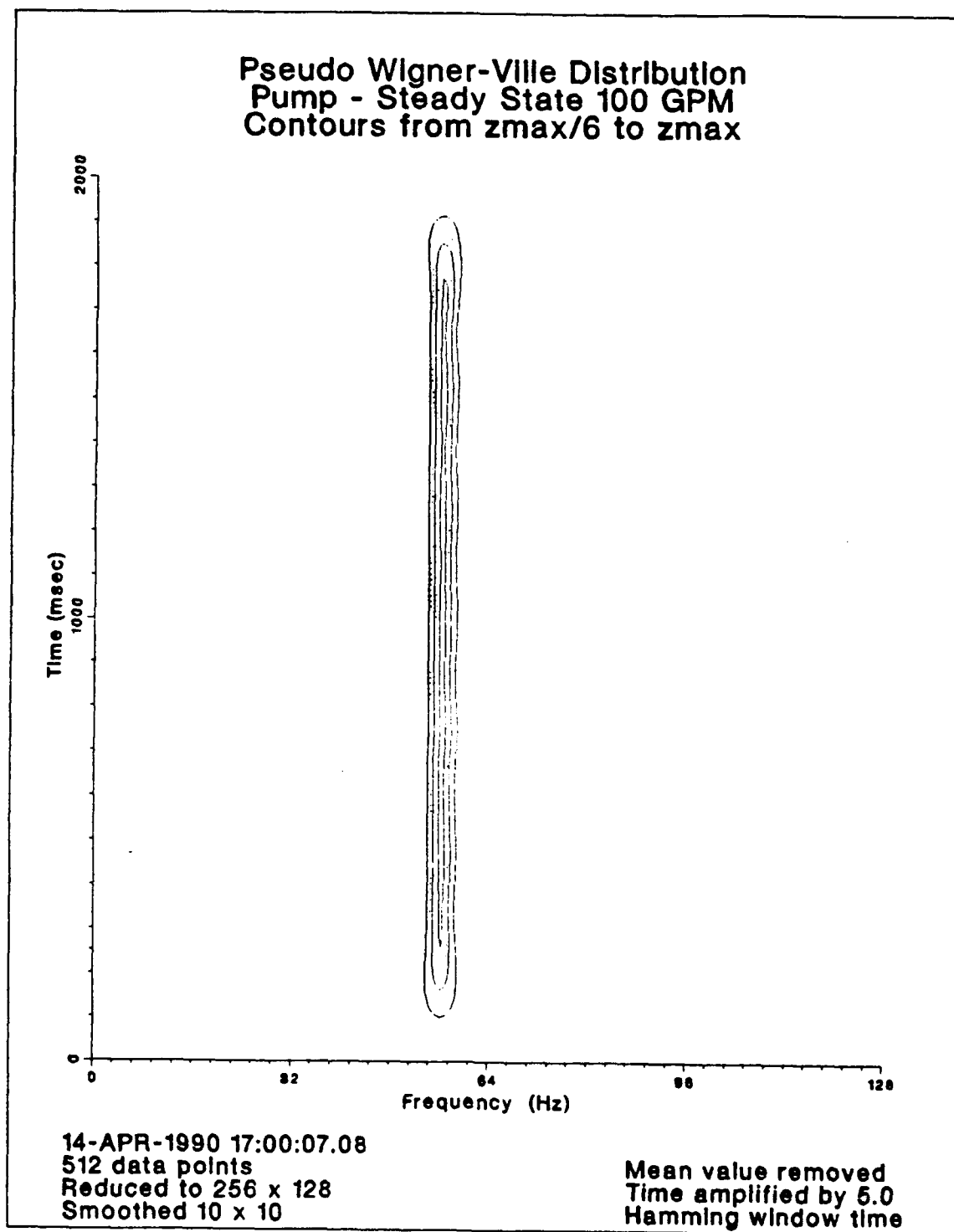
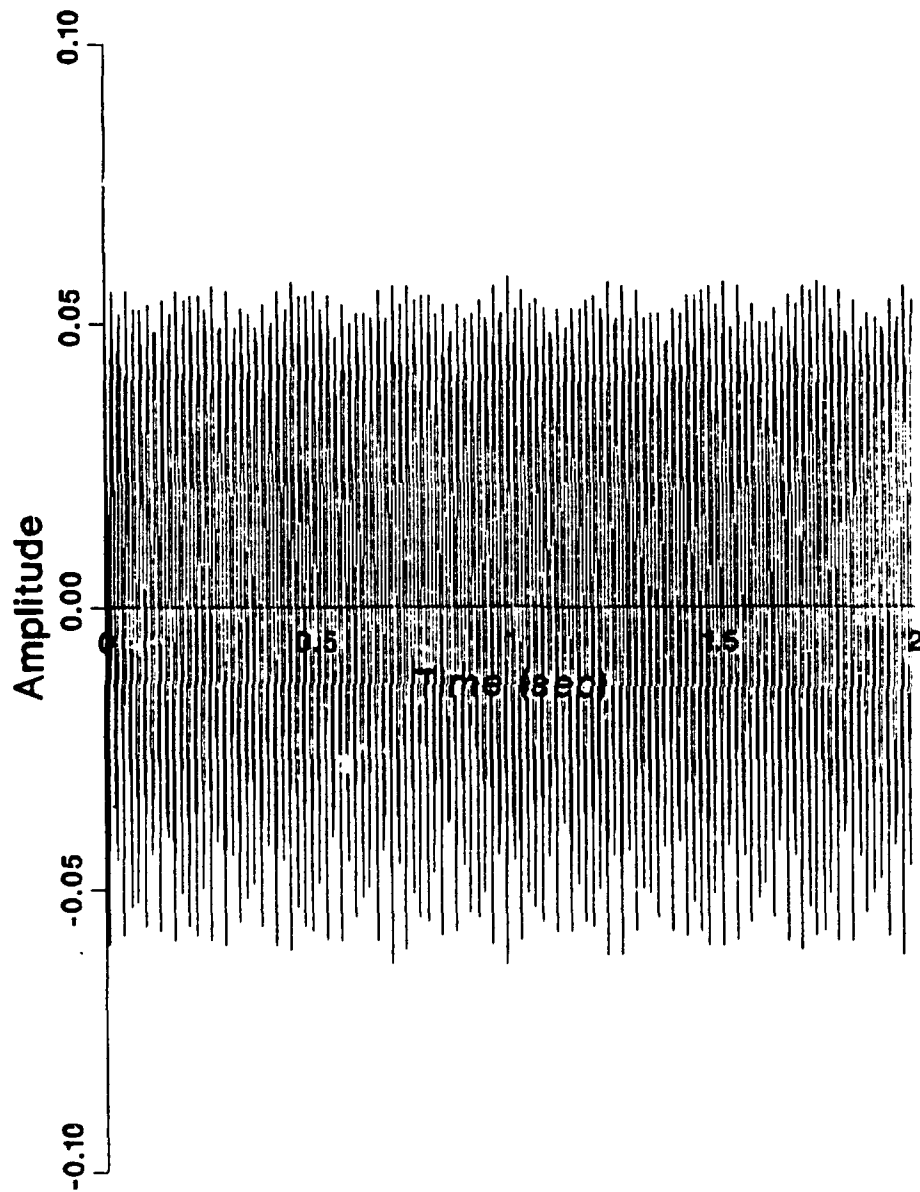


Figure 50. Detailed Contours of Steady State 100 GPM

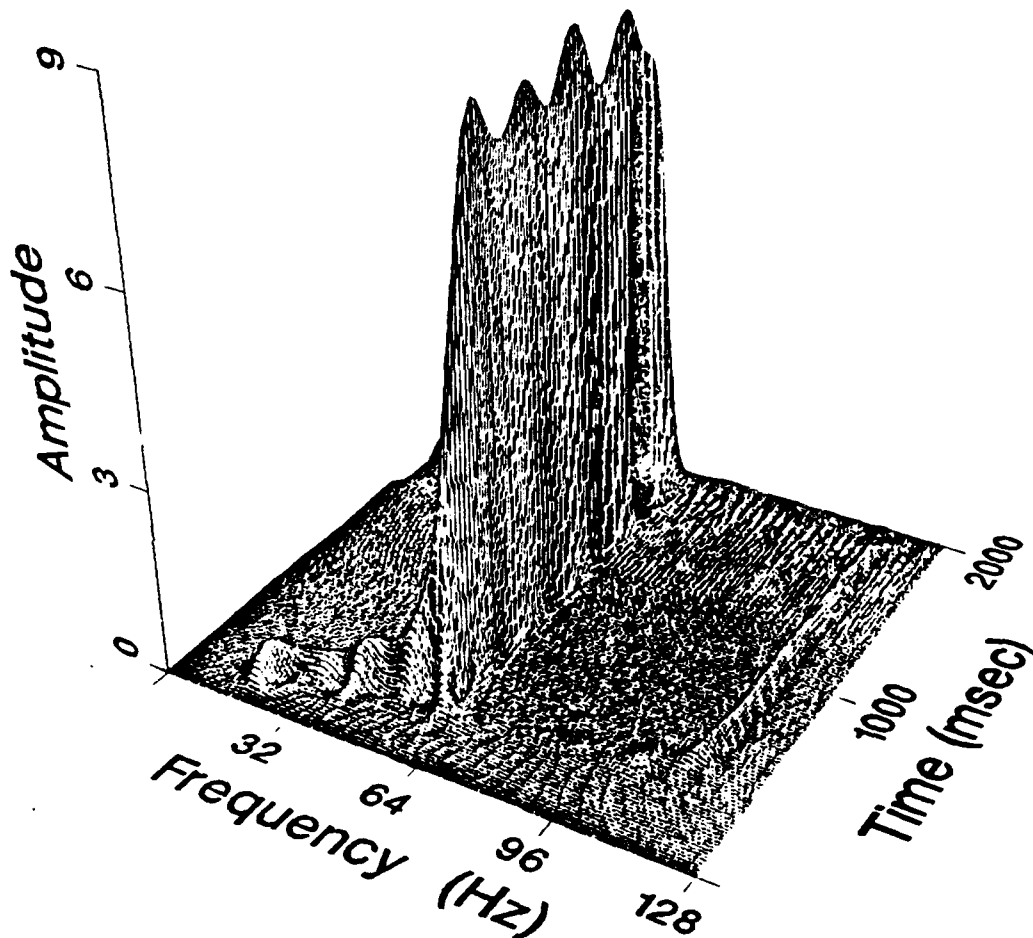
Raw Time Signal Pump - Steady State 100 GPM



14-APR-1990 14:49:51.63

Figure 51. Input Time Signal Steady State 100 GPM

Pseudo Wigner-Ville Distribution Pump Speed - Transient



14-APR-1990 17:16:26.47
512 data points
Reduced to 256 x 128
Smoothed 10 x 10

Mean value removed
Time amplified by 5.0
Hamming window time

Figure 52. Pump Transient Speed Up

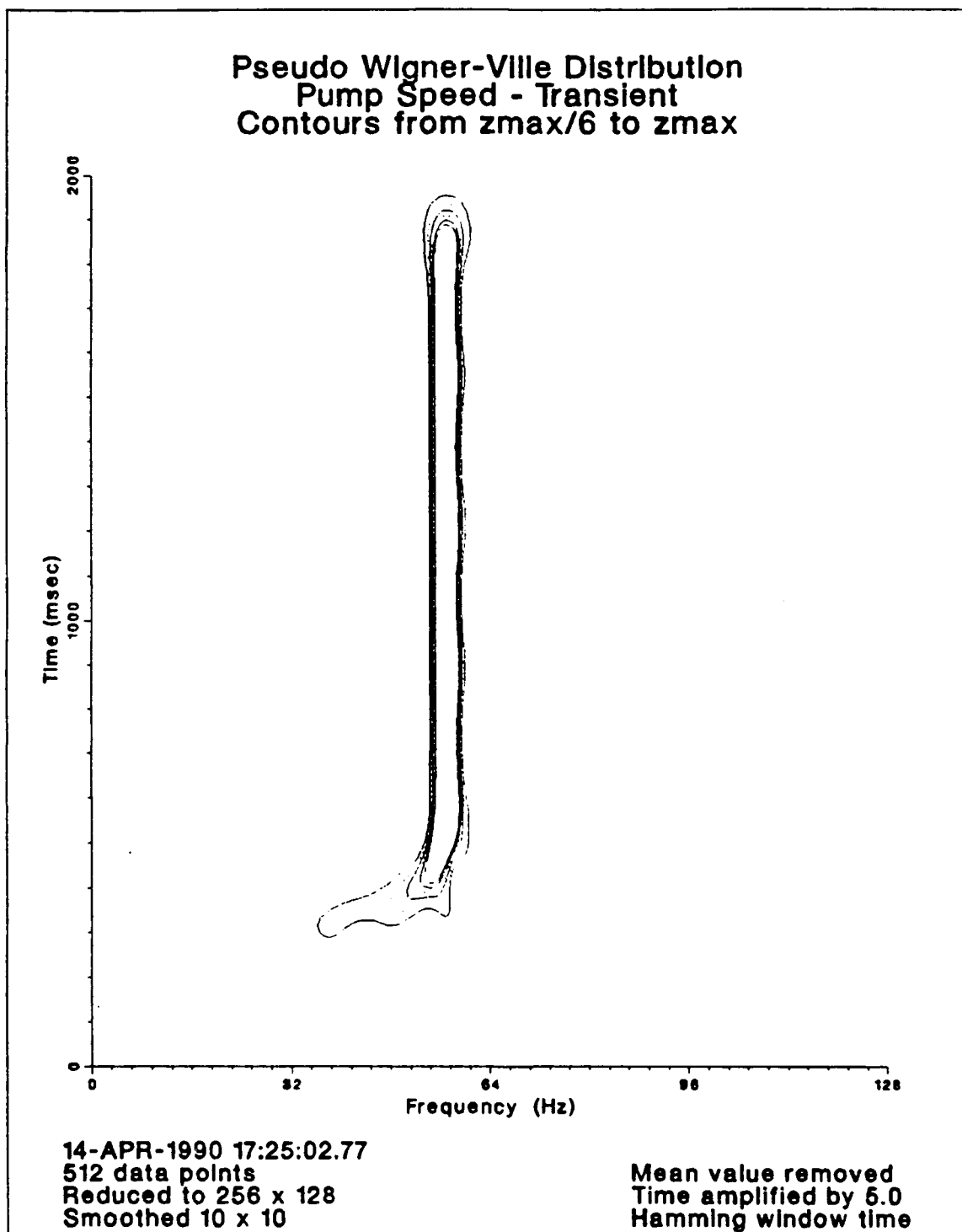
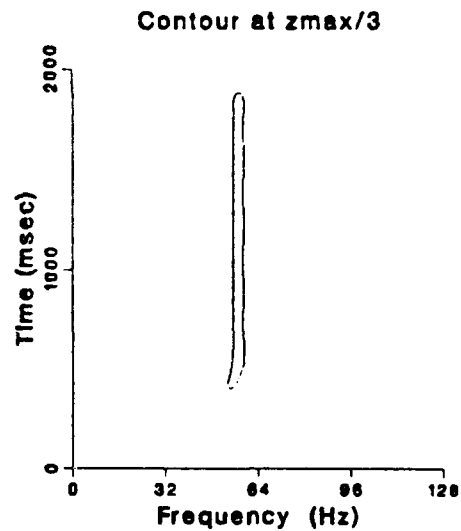
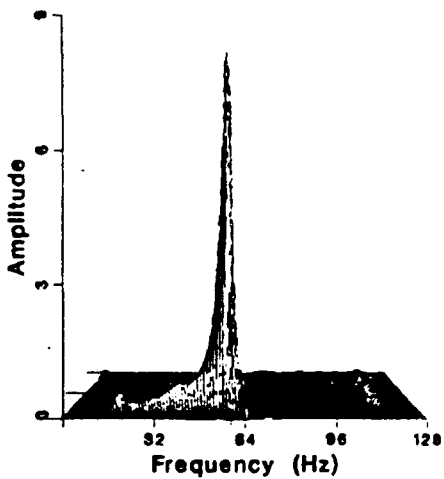
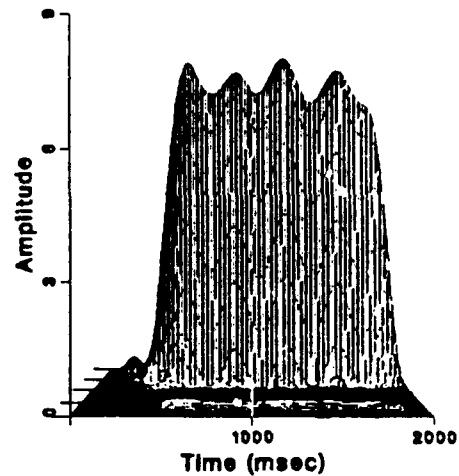
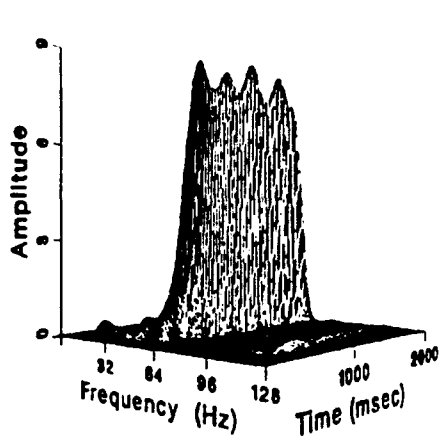


Figure 53. Detailed Contours of Speed Up

Pseudo Wigner-Ville Distribution Pump Speed - Transient



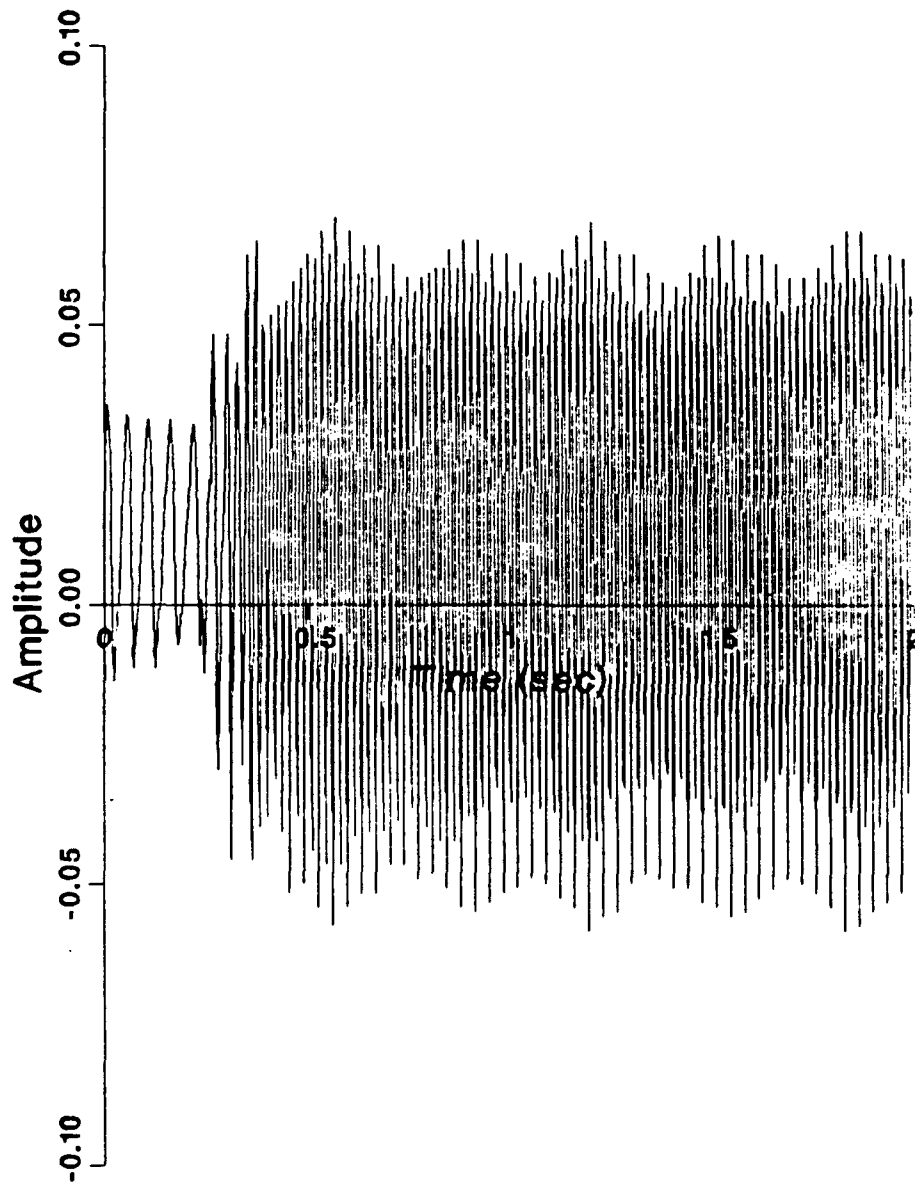
Contour at zmax/3

14-APR-1990 17:02:50.42
512 data points
Reduced to 256 x 128
Smoothed 10 x 10

Mean value removed
Time amplified by 5.0
Hamming window time

Figure 54. Multiple Views of Speed Up

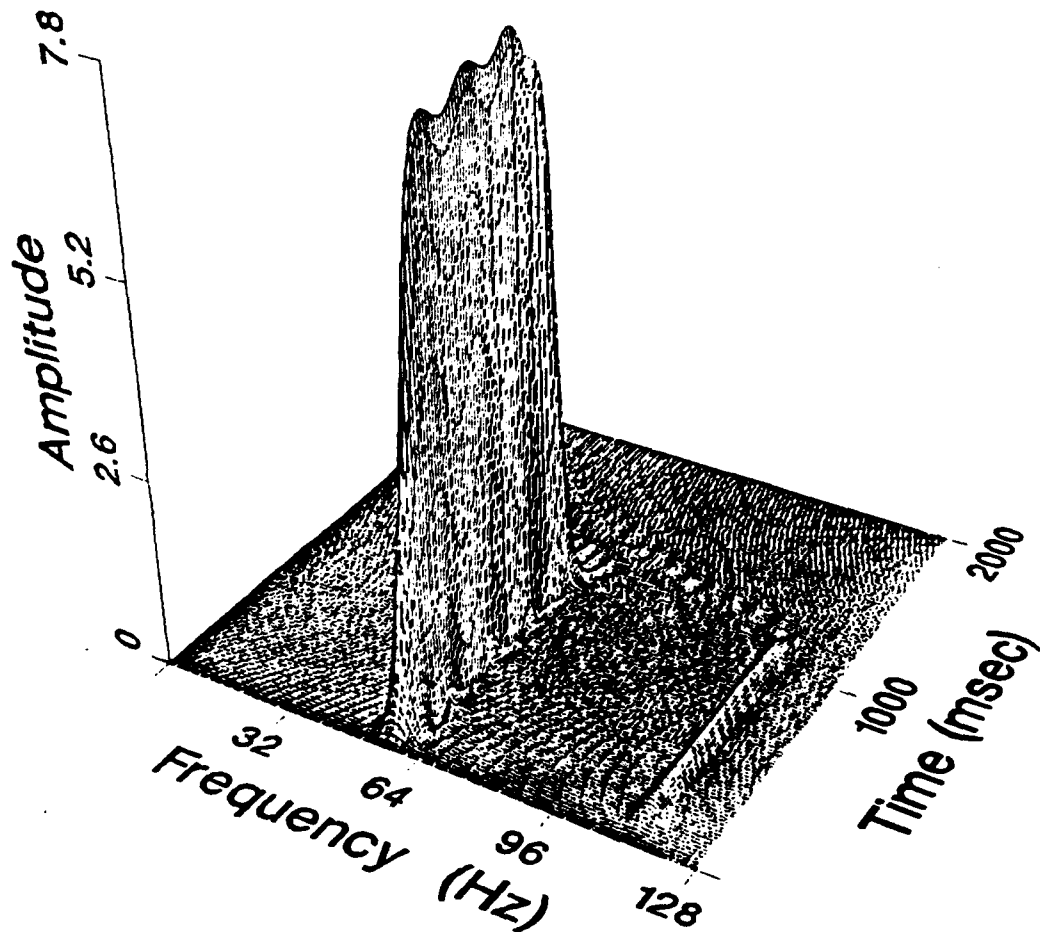
Raw Time Signal Pump Speed - Transient



14-APR-1990 15:20:59.50

Figure 55. Input Time Signal of Speed Up

Pseudo Wigner-Ville Distribution Pump Speed - Transient



14-APR-1990 18:42:09.83
512 data points
Reduced to 256 x 128
Smoothed 10 x 10

Mean value removed
Time amplified by 5.0
Hamming window time

Figure 56. Pump Transient Coast Down

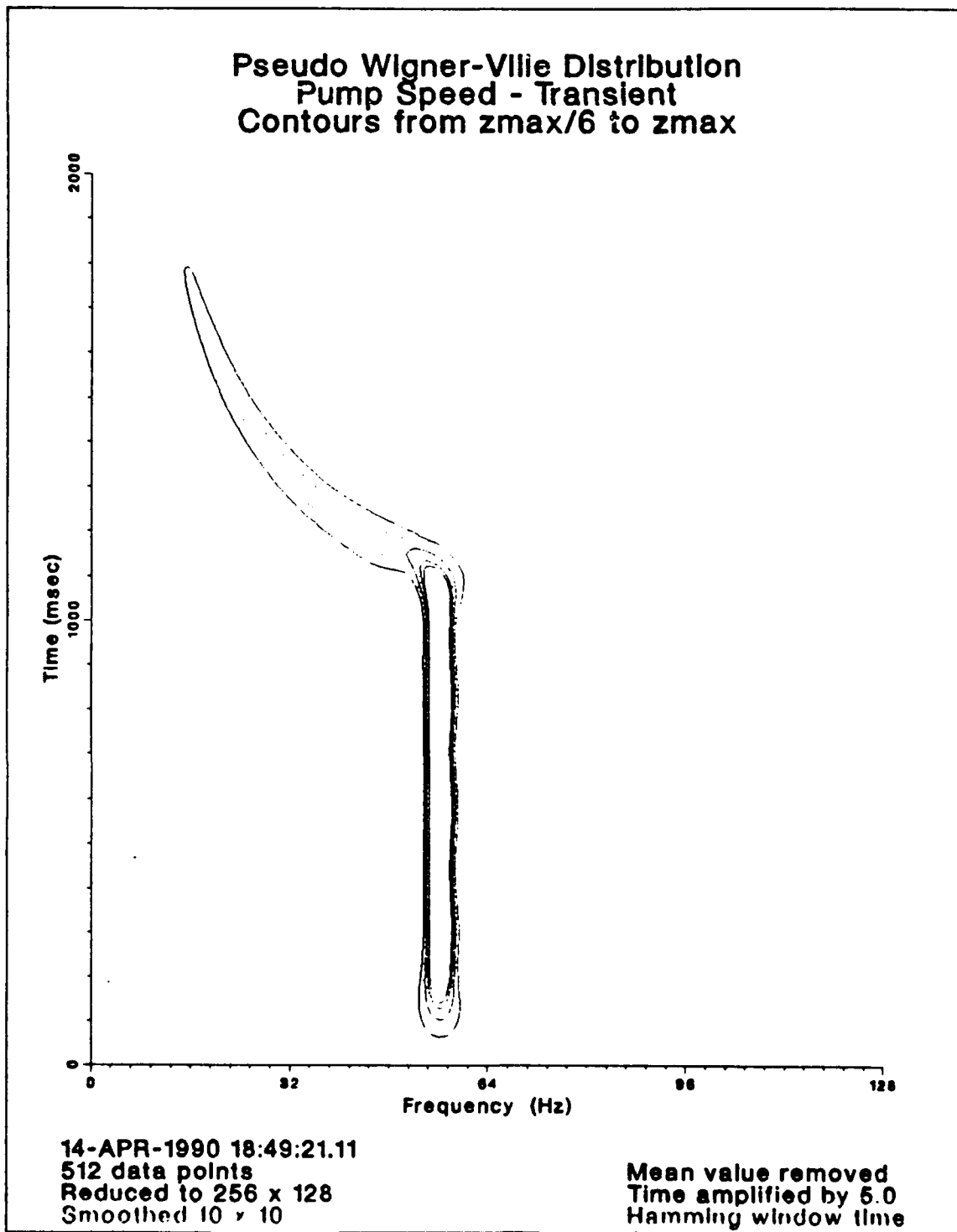


Figure 57. Detailed Contours of Coast Down

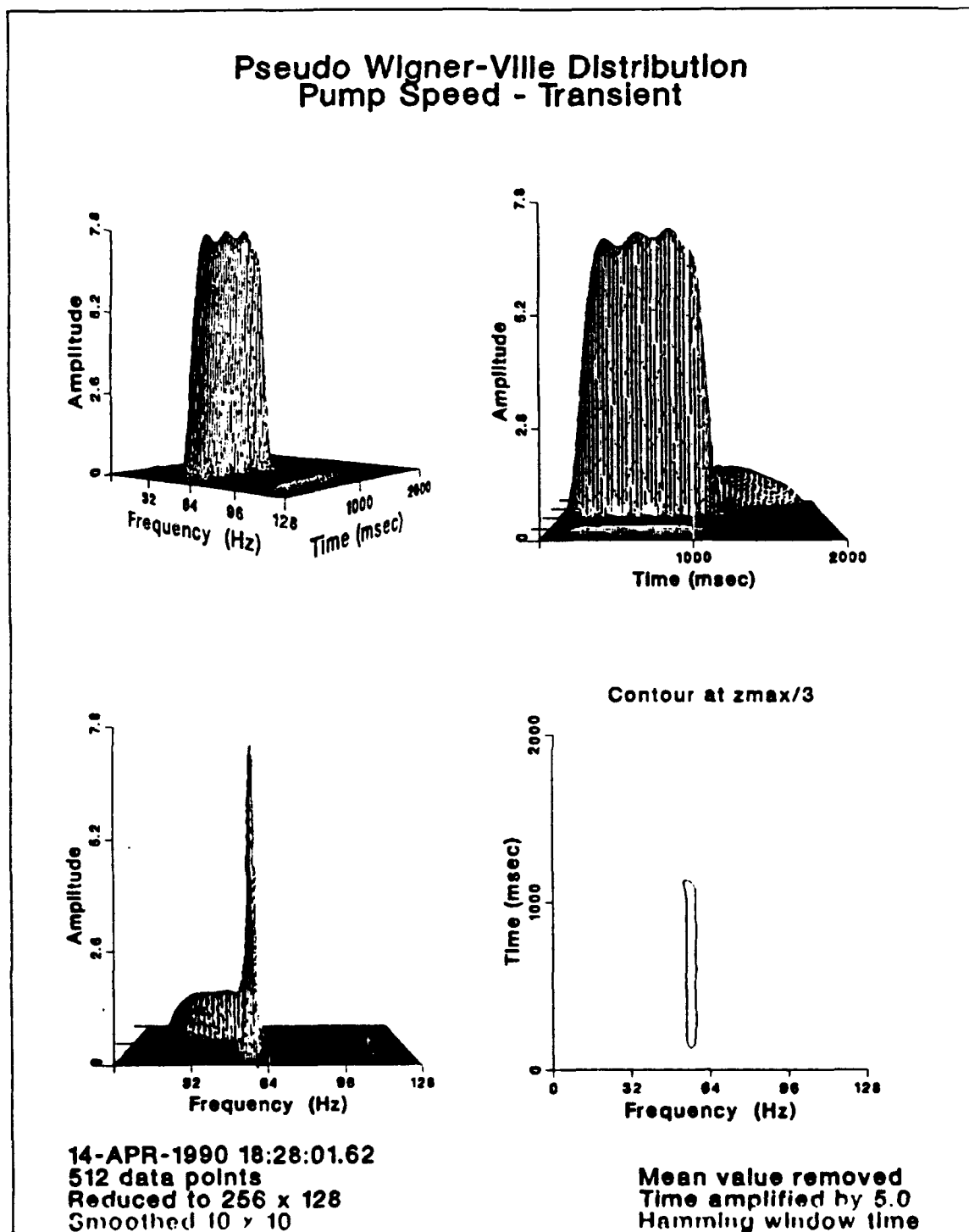
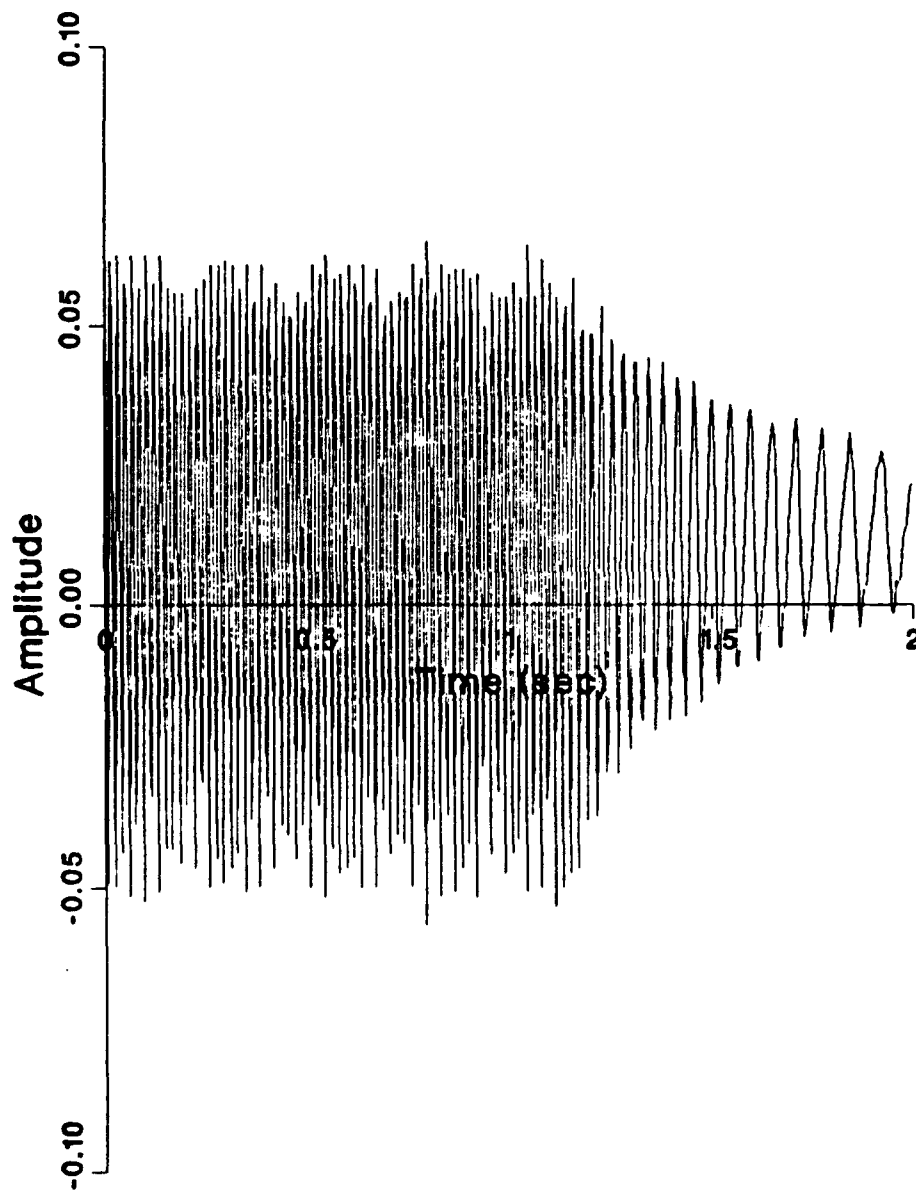


Figure 58. Multiple Views of Coast Down

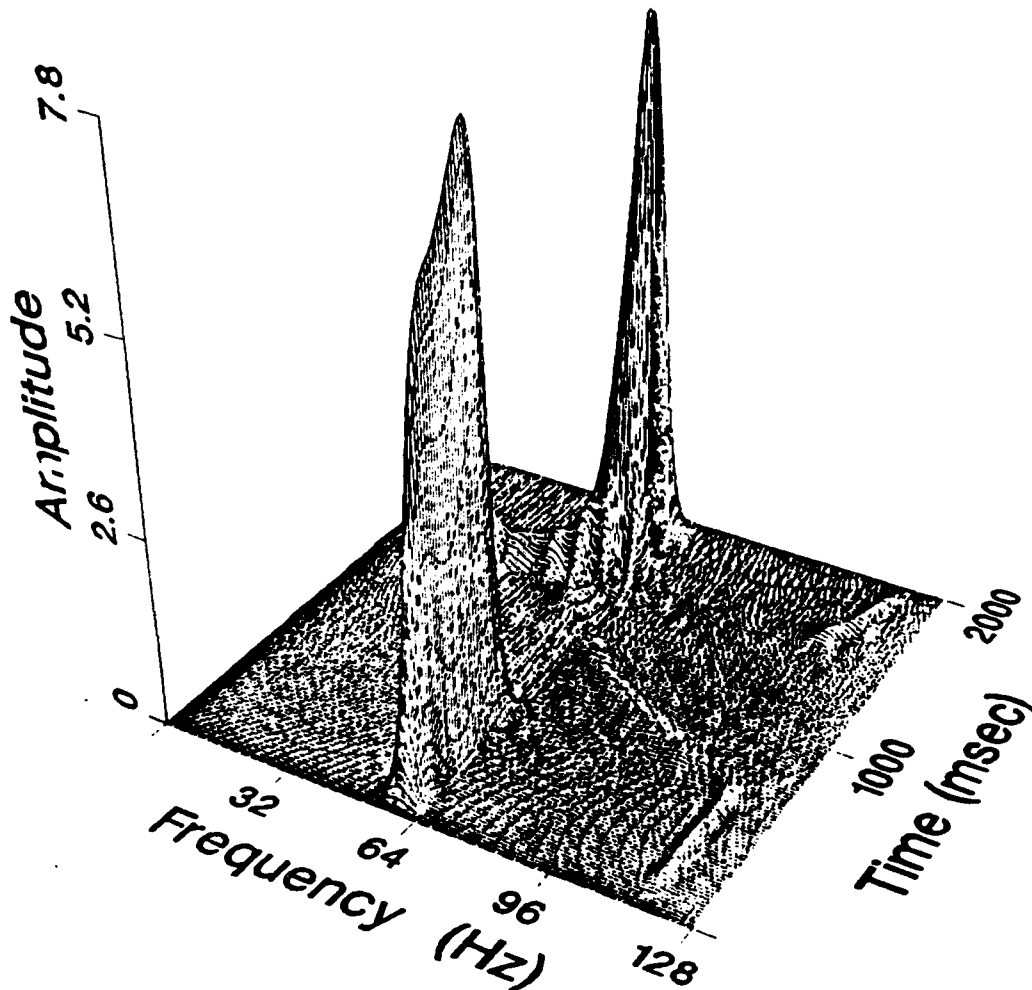
Raw Time Signal Pump Speed - Transient



14-APR-1990 17:33:13.51

Figure 59. Input Time Signal of Coast Down

Pseudo Wigner-Ville Distribution Pump Speed - Transient



14-APR-1990 17:10:26.30
512 data points
Reduced to 256 x 128
Smoothed 10 x 10

Mean value removed
Time amplified by 5.0
Hamming window time

Figure 60. Pump Transient Coast Down and Speed Up

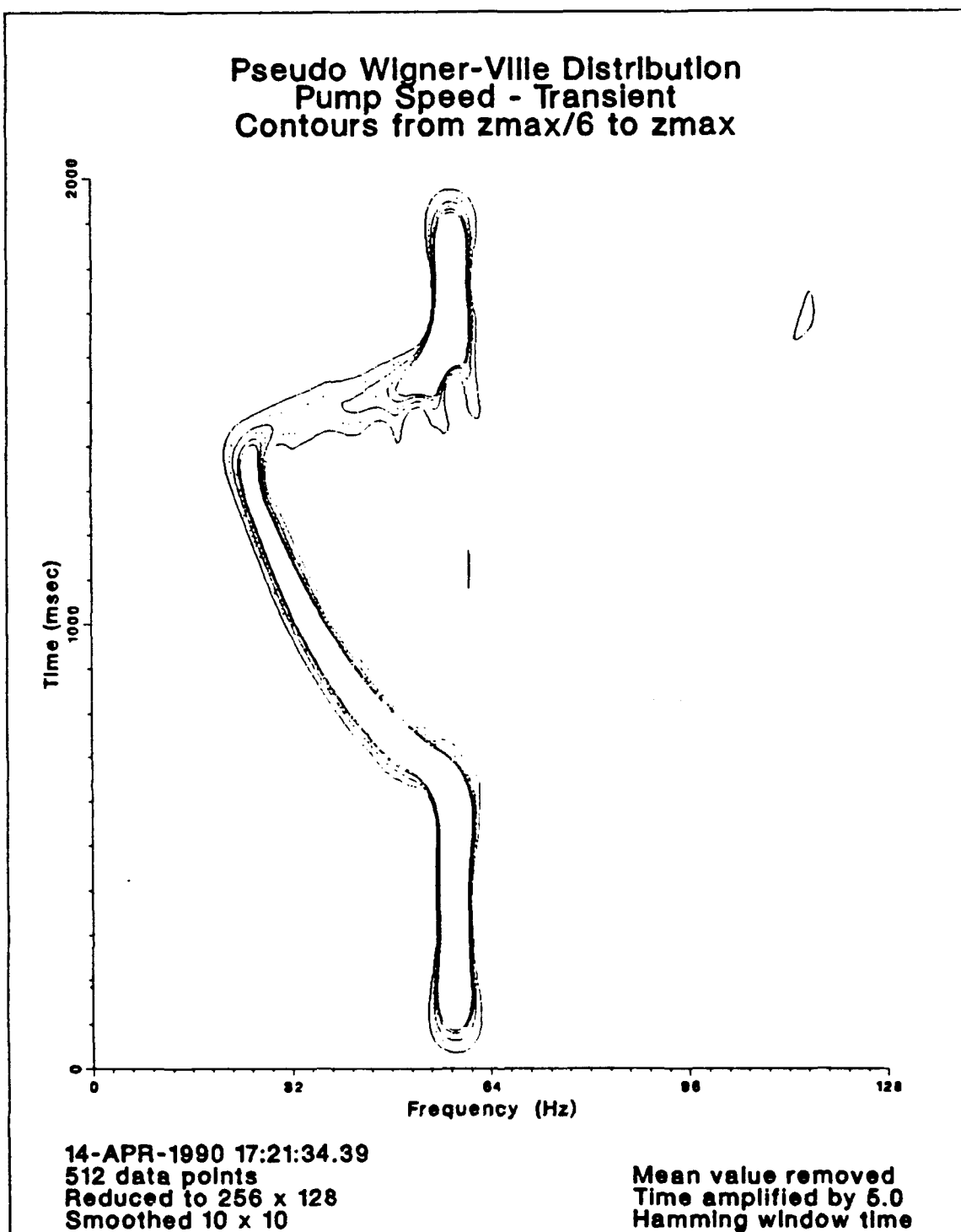
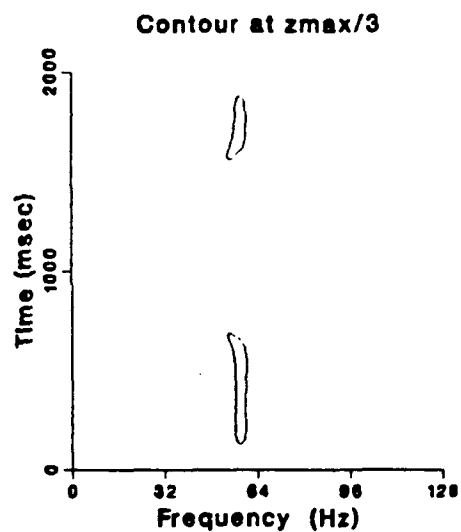
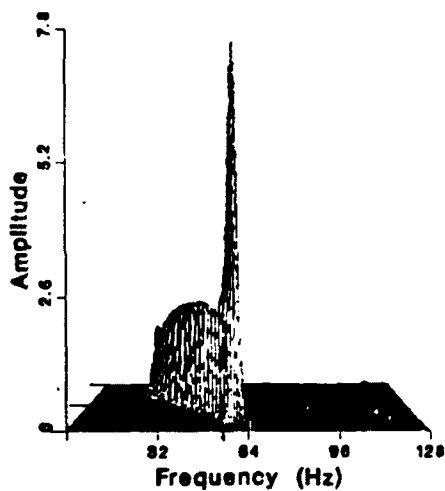
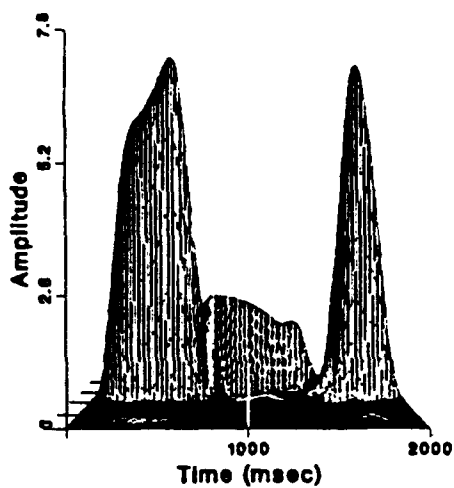
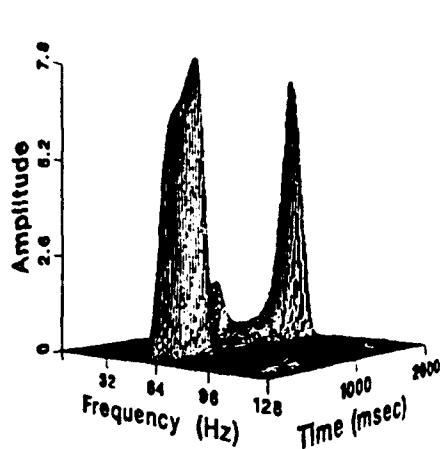


Figure 61. Detailed Contours of Coast Down and Speed Up

Pseudo Wigner-Ville Distribution Pump Speed - Transient

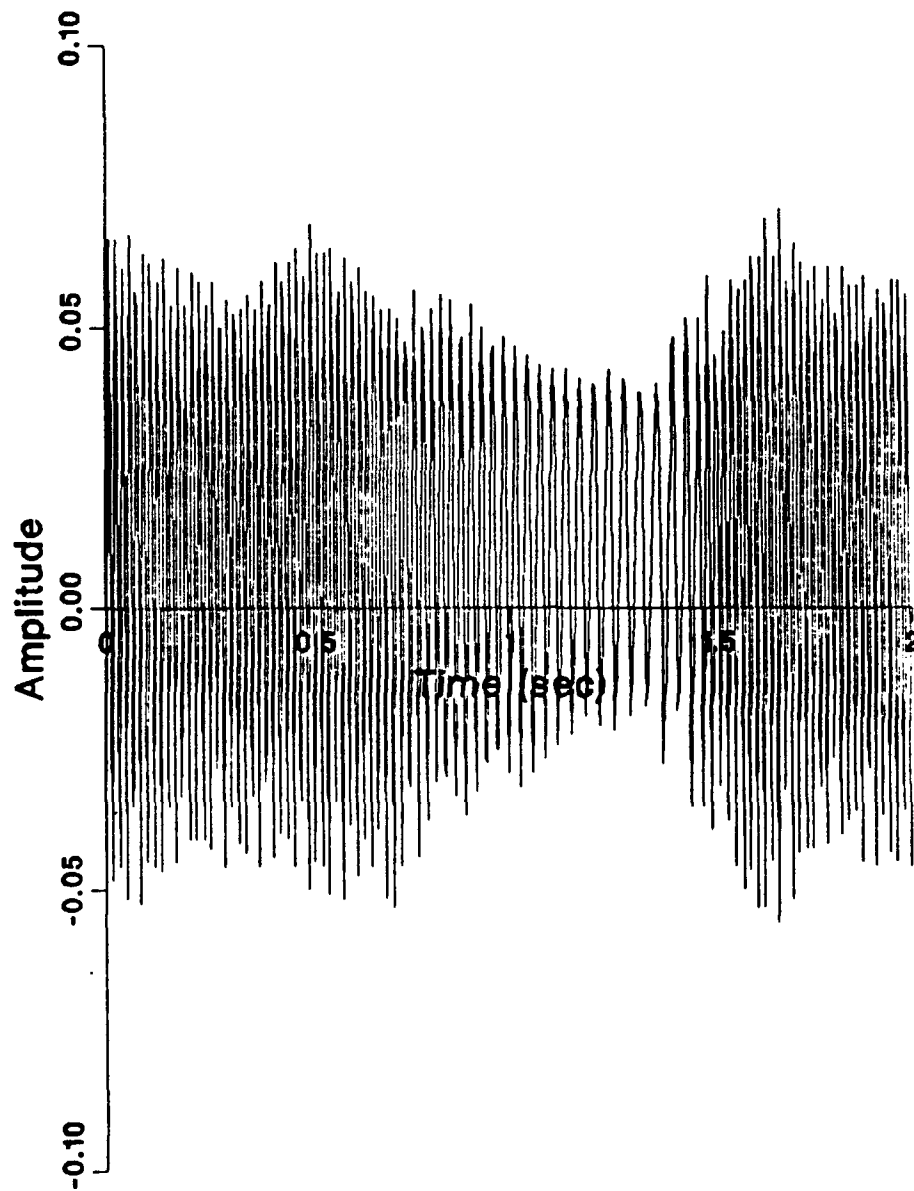


14-APR-1990 16:56:35.43
512 data points
Reduced to 256 x 128
Smoothed 10 x 10

Mean value removed
Time amplified by 5.0
Hamming window time

Figure 62. Multiple Views of Coast Down and Speed Up

Raw Time Signal Pump Speed - Transient



14-APR-1990 15:14:24.75

Figure 63. Input Time Signal of Coast Down and Speed Up

V. CONCLUSIONS

The Pseudo Wigner-Ville Distribution has been applied to analyzing data for use in machinery condition monitoring and diagnostics. From the research conducted it will be a valuable asset for analyzing transient machinery. The following conclusions can be drawn:

- The Pseudo Wigner-Ville Distribution is ideally suited for portraying non-stationary time signals.
- Time varying frequency components, as well as stationary frequency components, are evident in a single output.
- The use of an analytic signal in calculating the Pseudo Wigner-Ville Distribution eliminates aliased frequency components.
- The postprocessing smoothing process deemphasizes cross terms and provides a distribution directly related to the input signal.
- The amplitude of the Pseudo Wigner-Ville Distribution varies with noise and with changing frequencies over time.
- A mean value of an input time domain signal is a DC (0 Hz) component in the Pseudo Wigner-Ville Distribution.
- A modified Hamming window applied to an input time domain signal starts and ends the signal at zero amplitude and minimizes the distortion of the Pseudo Wigner-Ville Distribution.
- A time domain signal amplitude amplification, or gain, may be necessary to minimize roundoff error in calculating the Pseudo Wigner-Ville Distribution.

VI. RECOMMENDATIONS FOR FUTURE RESEARCH

The computer program works well and the Pseudo Wigner-Ville Distribution will be a valuable asset to assist in machinery condition monitoring. Improvements to the computer program could include:

- Increasing the number of data points being processed
- Including a digital filter
- Optimizing the routines used in order to increase the speed of calculations
- Making the Pseudo Wigner-Ville Distribution an accurate signal energy representation

Areas where additional research is needed include:

- Investigating the effect of using just the real part of the Wigner-Ville Distribution versus the real and imaginary parts or combinations thereof
- Investigating other sliding averaging windows in the smoothing routine
- Investigating the cause of the pronounced peak amplitudes evident in the swept sine wave examples

Recommendations for application to machinery condition monitoring include:

- Using an analog filter on all input data to ensure unwanted frequencies will not alias the frequencies of interest

APPENDIX A. COMPUTER CODE

A. SUMMARY

```
c
c      [rossano.thesis]symbols.include
c
c      This is a description of the programs, subroutines, and symbols
c      used to calculate the Pseudo Wigner-Ville Distribution
c
c  FORTRAN PROGRAMS
c
c      WIGFUN1
c          Reads in the real time signal, manipulates it, and
c          writes the resulting analytic signal into a file
c      WIGFUN1b
c          Plots the analytic time signal
c      WIGFUN2
c          Reads in the analytic time signal, calculates the
c          Wigner Distribution, and writes it to a file
c      WIGFUN3
c          Reads in the raw Wigner Distribution, manipulates
c          it by reducing and smoothing, and writes the results
c          to files
c      WIGFUN4a
c          Plots the 3 dimensional distributions which were
c          reduced to (64,32)
c      WIGFUN4b
c          Plots the 3 dimensional distributions which were
c          reduced to (128,64)
c      WIGFUN4c
c          Plots the 3 dimensional distributions which were
c          reduced to (256,128)
c
c  SUBROUTINES
c
c      AMPLIFY
c          Amplifies the time signal
c      ANALYTIC
c          Converts a real signal into an analytic one
c      CORR
c          Calculates the correlation
c      DATAIN
c          Reads the time signal data into an array
c      DATAOUT
c          Writes the WDF array to a file
c      DATAOUT2
c          Writes the RWDF and PWDF arrays to files
c      DTCALC
c          Calculates delta t from an input file or array
c      FFT
c          Calculates the Fast Fourier Transform
```

c HAMMING Applies a hamming window to a signal
 c HANNING Applies a hanning window to a signal
 c MAXAMP Finds the maximum amplitude of a signal
 c MEAN Calculates the mean value of a signal
 c MEANR Removes the mean value from a signal
 c MINAMP Finds the minimum amplitude of a signal
 c PLOT2D Plots one 2 dimensional curve
 c PLOT2D2 Plots two 2 dimensional curves on one plot
 c PLOT3D32 Plots one 3 dimensional graph of rwdf(64,32)
 c PLOT3D64 Plots one 3 dimensional graph of rwdf(128,64)
 c PLOT3D128 Plots one 3 dimensional graph of rwdf(256,128)
 c PLOT3DSPLIT32 Plots four 3 dimensional graphs of rwdf(64,32)
 c in a split screen format each with a different
 c viewing perspective
 c PLOT3DSPLIT64 Plots four 3 dimensional graphs of rwdf(128,64)
 c in a split screen format each with a different
 c viewing perspective
 c PLOT3DSPLIT128 Plots four 3 dimensional graphs of rwdf(256,128)
 c in a split screen format each with a different
 c viewing perspective
 c PLOTCON32 Plots several levels on a single contour plot for
 c rwdf(64,32)
 c PLOTCON64 Plots several levels on a single contour plot for
 c rwdf(128,64)
 c PLOTCON128 Plots several levels on a single contour plot for
 c rwdf(256,128)
 c PSEUDO Smooths the WDF along both the time and frequency axes
 c RANGE32 Finds the maximum and minimum amplitudes of
 c rwdf(64,32)
 c RANGE64 Finds the maximum and minimum amplitudes of
 c rwdf(128,64)
 c RANGE128 Finds the maximum and minimum amplitudes of
 c rwdf(256,128)
 c REDUCE Reduces the data in WDF to RWDF
 c

```

c      SIZE
c      Changes array sizes for time plotting routines
c      TIMESIG
c      Modifies and plots the time signal
c      WIGH
c      Calculates the WDF
c      WINDOW
c      Calls the available windowing functions
c
c  SYMBOLS
c
c      ain(j)=signal amplitude array
c      amp=input data amplitude
c      ampl=amplification applied to a signal
c      amplif=amplification information for plot
c      anq=answer to analytic question
c      atvq=answer to various questions
c      avgdelt=average dt found in input time array
c      c(j)=complex correlation array
c      ci(j)=imaginary correlation data array
c      coef=coefficient used in mathematical calculations
c      const=constant used in mathematical calculations
c      cr(j)=real correlation data array
c      datetime=date and time obtained from the computer
c      delt(j)=array of actual dt values
c      delta=a time increment used in calculating hamming window
c      df=frequency step
c      dimf=dimension size for frequency
c      dimt=dimension size for time
c      dp=number of data points
c      dp2=dp*2
c      dpend=modification of dp in input array gathering to ensure
c           that dp number of points will be collected
c      dpnum=number of data point information for plot
c      dt=time step
c      dum=complex variable used in mathematical calculations
c      dum2=complex variable used in mathematical calculations
c      dum3=complex variable used in mathematical calculations
c      hg(i,j)=smoothing multiplier
c      i=a counter
c      ii=a counter
c      inname=input file name
c      inv=indicator for fft or inverse fft
c      iplot_val=answer whether graph output on screen or hardcopy
c      j=a counter
c      jj=a counter
c      k=a counter
c      kk=a counter
c      L=a counter
c      LL=a counter
c      label=instructions for renaming std00001.dat
c      lfile=plotting file name
c      meani=mean information for plot
c      meanv=mean value of a signal
c      mesh=mesh size for plotting

```

```

c      mfreq=max frequency (Hz)
c      mm=a variable used for reducing the number of time points in
c          pwdf and rwdf plots
c      mnq=answer for mean question
c      mt=number of points smoothed, time
c      mt2=mt*2
c      mtime=max time (sec) NOTE: converted to (msec) for pwdf and
c          rwdf plots
c      n=a counter
c      nf=number of points smoothed, frequency
c      nf2=nf*2
c      nn=a variable used for reducing the number of frequency points
c          in pwdf and rwdf plots
c      outname=output file name
c      pi=mathematical quantity, pi
c      pi2=2*pi
c      pwdf(i,j)=pseudo wigner distribution
c      rcnum=variable for which correlation to plot
c      rdp=reduced number of data points
c      rdp2=rdp*2
c      rnq=answer for rename std00001.dat question
c      rval=reduction value, indicates number of data points
c          reduced to
c      rwdf(i,j)=reduced wigner distribution
c      s(j)=complex time array
c      si(j)=imaginary time data array
c      sizedp=maximum size available, set by dimension of arrays
c      sr(j)=real time data array
c      ss=step size used to modify dt thereby changing max freq
c      starttime=a holder for datetime
c      status=variable used with datetime
c      sum=sum of values
c      sumb=sum of values
c      sval=sine of val
c      t(j)=data array
c      tcode=time window code
c      tim=input data time
c      tin(j)=signal time array
c      title=title of a plot
c      titlehold=holds 3d plot title name while plotting correlation
c      val=value used for mathematical calculations
c      wdf(i,j)=raw wigner distribution
c      windw=value of window function at a point
c      wndwcode=type of window code
c      x(j)=data array
c      xmax=maximum amplitude
c      xmin=minimum amplitude
c      y(j)=data array
c      ymax=maximum amplitude
c      ymin=minimum amplitude
c      zhold=holder for calculating zmax or zmin
c      zmax=maximum amplitude
c      zmin=minimum amplitude
c
c      NOTE: The plotting routines use CA-DISSPLA Version 11.0

```

c written by Computer Associates International, Inc.
c

B. FORTRAN PROGRAMS

c [rossano.thesis]wigfun1.for
c
c Graham W. Rossano
c Prof. J. Hamilton
c Prof. Y.S. Shin
c
c This program is the first of four that calculate and plot the
c Wigner Distribution Function and variations of a signal
c
c WIGFUN1 reads in the real time signal, manipulates it, and writes
c the resulting analytic signal into a file
c
c All of the subroutines have been broken off for ease of modifying
c and printing. Upon compiling, they are all included.
c
c This program uses an include statement to facilitate changing
c the size of the plotting arrays.
c (see [rossano.thesis]size.include)
c
c A description of the symbols used in this program and its
c subroutines may be found in [rossano.thesis]symbols.include
c

```
complex s(1025)
real ain(1025),mtime,mfreq,avgdelt,dt,sr(1025),si(1025),
&tin(1025),df,ampl
character*23 datetime,tcode
character*25 inname,lfile
character*50 title,outname
character*80 label
integer*4 status
integer dp,dimt,dimf,rnq,dp2,j,mnq
include 'size.include'
```

c Start time of program
status = lib\$date_time (datetime)
print*
type *, datetime

c Filename assignments
print*
print*, ' Program to calculate and plot'
print*, ' Wigner Distribution Function'
print*
print*, ' WIGFUN1'
print*
print*, ' Enter signal input filename'
read(5,904) inname
print*
print*, ' Enter plotting output filename'
read(5,901) outname

```

        print*

c Calculations
    dimt=1025
    call dtcalc (dimt,1,inname,tin,avgdelt)
    print*, ' Average Delta t ='
    write(5,906) avgdelt
    print*
    print*, ' Input the number of data points to evaluate'
    print*, '          (128, 256, 512, or 1024)'
    print*
    print*, ' This size depends on the size of the arrays set in: '
    print*, ' [rossano.thesis]size.include'
    print*
    print*, ' If you change it you will need to recompile and lnkdss'
100  read(5,*) dp
    print*
    if (dp.ne.sizedp) then
        print*, ' sizedp='
        print*, sizedp
        print*, ' Reenter dp'
        go to 100
    endif
    if (dp.eq.128) then
        dimt=130
        dimf=260
    endif
    if (dp.eq.256) then
        dimt=260
        dimf=520
    endif
    if (dp.eq.512) then
        dimt=520
        dimf=1040
    endif
    if (dp.eq.1024) then
        dimt=1025
        dimf=2050
    endif
    call datain (dimt,inname,avgdelt,dp,ain,dt)
    tcode='No window time'
    wndwcode=0
    print*, ' Input signal has been put into an array'
    print*
    call timesig (dimt,ain,dp,dt,outname,tcode,
&title,s,sr,si,mnq,ampl)
    open (unit=7,file='wig1.log',status='new')
    open (unit=8,file='wig1.dat',status='new')
    write(7,908) dimt,dimf,dp,dt
    write(7,905) outname,tcode,mnq
    write(7,909) title,ampl
    do 200 j=1,dp
        write(8,*) s(j)
200  continue
    close (unit=7)
    close (unit=8)

```

```

print*, ' Do you want to rename std00001.dat?'
print*, '      (1 for yes, 2 for no)'
read(5,*) rnq
print*
if (rnq.eq.1) then
  print*, ' Enter Laser plot file name'
  read(5,904) lfile
  LABEL(1:20)='RENAME STD00001.DAT '
  LABEL(21:45)=lfile
  CALL LIB$SPAWN(LABEL)
  print*
endif
print*, ' Analytic data is in wig1.dat'
print*, ' Plotting info is in wig1.log'
print*
print*, ' Now run WIGFUN2'
print*

```

c Format statements

```

901 format(a50)
904 format(a25)
905 format(2x,a50,2x,a23,2x,i8)
906 format(f16.8)
908 format(2x,i8,2x,i8,2x,i8,2x,e16.8)
909 format(2x,a50,2x,f16.8)
end

```

c SUBROUTINES

```

include 'AMPLIFY.INCLUDE'
include 'ANALYTIC.INCLUDE'
include 'DATAIN.INCLUDE'
include 'DTCALC.INCLUDE'
include 'HAMMING.INCLUDE'
include 'HANNING.INCLUDE'
include 'MAXAMP.INCLUDE'
include 'MEAN.INCLUDE'
include 'MEANR.INCLUDE'
include 'MINAMP.INCLUDE'
include 'PLOT2D.INCLUDE'
include 'PLOT2D2.INCLUDE'
include 'TIMESIG.INCLUDE'
include 'WINDOW.INCLUDE'

```

c

c [rossano.thesis]wigfun1b.for

c

c Graham W. Rossano

c Prof. J. Hamilton

c Prof. Y.S. Shin

c

c This program plots the analytic time signal

c

c All of the subroutines have been broken off for ease of modifying
c and printing. Upon compiling, they are all included.

c

c A description of the symbols used in this program and its

```

c      subroutines may be found in [rossano.thesis]symbols.include
c

complex s(550)
real mtime,mfreq,dt,sr(550),si(550),df,ampl
character*23 datetime,tcode
character*25 lfile
character*50 title,outname,titlehold
character*80 label
integer*4 status
integer dp,dimt,dimf,rnq,dp2,sizedp,mnq

c Start time of program
status = lib$date_time (datetime)
print*
type *, datetime
sizedp=512

c Get data from WIGFUN1
print*
print*, ' Program to plot the analytic time signal'
print*
print*, ' WIGFUN1b'
print*
open (unit=7,file='wig1.log',status='old')
open (unit=8,file='wig1.dat',status='old')
rewind 7
rewind 8
read(7,908) dimt,dimf,dp,dt
read(7,905) outname,tcode,mnq
read(7,909) title,ampl
do 200 j=1,dp
    read(8,*) s(j)
200 continue
close (unit=7)
close (unit=8)
print*, ' Data is loaded'
print*

c Calculations
if (dp.gt.sizedp) then
    print*, ' max number of points allowed is'
    print*,sizedp
    print*, ' dp='
    print*,dp
    print*
    print*, ' Check your dimensions'
    stop
endif
dp2=dp*2
mtime=dp*dt
df=1./(4.*mtime)
mfreq=2.*dp*df
do 100 j=1,dp
    sr(j)=real(s(j))
    si(j)=aimag(s(j))

```



```

100 continue
print*
print*, ' Do you want to plot the analytic time signal?'
print*, ' (1 for yes, 2 for no)'
read(5,*) ptaq
if (ptaq.eq.1) then
  titlehold=title
  title='Analytic Time Signal'
  call plot2d2 (sr,si,dt,dp,title,outname,mnq)
  title=titlehold
endif
print*, ' Do you want to rename std00001.dat?'
print*, ' (1 for yes, 2 for no)'
read(5,*) rnq
print*
if (rnq.eq.1) then
  print*, ' Enter Laser plot file name'
  read(5,904) lfile
  LABEL(1:20)='RENAME STD00001.DAT '
  LABEL(21:45)=lfile
  CALL LIB$SPAWN(LABEL)
endif
print*, ' WIGFUN1b.FOR is complete'

```

c Format statements

```

904 format(a25)
905 format(2x,a50,2x,a23,2x,i8)
908 format(2x,i8,2x,i8,2x,i8,2x,e16.8)
909 format(2x,a50,2x,f16.8)
end

```

c SUBROUTINES

```

include 'MAXAMP.INCLUDE'
include 'MINAMP.INCLUDE'
include 'PLOT2D2.INCLUDE'

```

```

c
c      [rossano.thesis]wigfun2.for
c
c      Graham W. Rossano
c      Prof. J. Hamilton
c      Prof. Y.S. Shin
c
c      This program is the 2nd of four to calculate and plot the
c      Wigner Distribution Function and variations of a signal
c
c      WIGFUN2 reads in the analytic time signal from a file,
c      calculates the Wigner Distribution Function, and writes the
c      array to a file.
c
c      All of the subroutines have been broken off for ease of modifying
c      and printing. Upon compiling, they are all included.
c
c      A description of the symbols used in this program and its
c      subroutines may be found in [rossano.thesis]symbols.include
c

```

```

        complex s(550),c(1100)
        real mtime,mfreq,dt,cr(1100),ci(1100),df,
&wdf(1100,550),ampl
        character*23 datetime,tcode
        character*50 title,outname
        integer*4 status
        integer dp,dimt,dimf,dp2,sizedp,mnq
        common /wdfc/ wdf

c Start time of program
        status = lib$date_time (datetime)
        print*
        type *, datetime
        sizedp=512

c Get data from WIGFUN1
        print*
        print*, ' Program to calculate and plot'
        print*, ' Wigner Distribution Function'
        print*
        print*, ' WIGFUN2'
        print*
        open (unit=7,file='wig1.log',status='old')
        open (unit=8,file='wig1.dat',status='old')
        rewind 7
        rewind 8
        read(7,908) dimt,dimf,dp,dt
        read(7,905) outname,tcode,mnq
        read(7,909) title,ampl
        do 200 j=1,dp
            read(8,*) s(j)
200    continue
        close (unit=7)
        close (unit=8)
        print*, ' Data is loaded'
        print*

c Calculations
        if (dp.gt.sizedp) then
            print*, ' max number of points allowed is'
            print*, sizedp
            print*, ' dp='
            print*, dp
            print*
            print*, ' Check your dimensions'
            stop
        endif
        dp2=dp*2
        mtime=dp*dt
        df=1./(4.*mtime)
        mfreq=2.*dp*df
        call wigh (dimt,dimf,s,c,df,dt,dp2,dp)
        call dataout (dp)
        print*
        print*, ' Now run WIGFUN3'

```

```

        print*

c Format statements
905  format(2x,a50,2x,a23,2x,i8)
908  format(2x,i8,2x,i8,2x,i8,2x,e16.8)
909  format(2x,a50,2x,f16.8)
        end

c SUBROUTINES

        include 'CORR. INCLUDE'
        include 'DATAOUT. INCLUDE'
        include 'FFT. INCLUDE'
        include 'WIGH. INCLUDE'

c
c      [rossano.thesis]wigfun3.for
c
c      Graham W. Rossano
c      Prof. J. Hamilton
c      Prof. Y. S. Shin
c
c      This program is the 3rd of four to calculate and plot the
c      Wigner Distribution Function and variations of a signal
c
c      WIGFUN3 reads in the raw Wigner Distribution, manipulates
c      it by reducing and smoothing, and writes the resulting
c      arrays to files
c
c      All of the subroutines have been broken off for ease of modifying
c      and printing. Upon compiling, they are all included.
c
c      A description of the symbols used in this program and its
c      subroutines may be found in [rossano.thesis]symbols.include
c

        real mtime,mfreq,dt,df,
&wdf(1100,550),rwdf(256,128),ampl
        character*23 datetime,tcode
        character*25 inname
        character*50 title,outname
        integer*4 status
        integer dp,dimt,dimf,dp2,mnq,nn,mm,sizedp,rval
        common /wdfc/ wdf

c Start time of program
        status = lib$date_time (datetime)
        print*
        type *, datetime
        sizedp=512
        nn=1
        mm=1

c Get data from WIGFUN2
        print*
        print*, ' Program to calculate and plot'

```

```

print*, ' Wigner Distribution Function'
print*
print*, ' WIGFUN3'
print*
open (unit=7,file='wig1.log',status='old')
rewind 7
read(7,908) dimt,dimf,dp,dt
read(7,905) outname,tcode,mnq
read(7,909) title,ampl
close (unit=7)
print*, ' Wig1.log is loaded'
print*

```

c Calculations

```

      if (dp.gt.sizedp) then
        print*, ' max number of points allowed is'
        print*, sizedp
        print*, ' dp='
        print*, dp
        print*
        print*, ' Check your dimensions'
        stop
      endif
      dp2=dp*2
      mtime=dp*dt
      df=1./(4.*mtime)
      mfreq=2.*dp*df
      open (unit=13,file='wdf.unf',status='old',
&form='unformatted')
      rewind 13
      print*, ' reading wdf from wdf.unf'
      print*
      do 100 j=1,dp
        do 200 i=1,dp2
          read(13) wdf(i,j)
200    continue
100  continue
      close (unit=13)
      print*, ' wdf array is loaded'
      print*
      call reduce (dimt,dimf,dt,dp,nn,mm,rwdf,rval)
      print*
      call dataout2 (rwdf,3,dp,nn,mm)
      print*
      call pseudo (dimt,dimf,rwdf,dp,dt,nn,mm)
      print*
      call dataout2 (rwdf,2,dp,nn,mm)
      print*
      if (title.eq. 'Wigner Distribution') then
        title='Pseudo Wigner Distribution'
      endif
      if (title.eq. 'Wigner-Ville Distribution') then
        title='Pseudo Wigner-Ville Distribution'
      endif
      open (unit=7,file='wig3.log',status='new')
      write(7,908) dimt,dimf,dp,dt

```

```

write(7,905) outname,tcode,mnq
write(7,907) title,rval,ampl
close (unit=7)
print*, ' Updated plotting info is in wig3.log'
print*
if (rval.eq.1) print*, ' Now run WIGFUN4a'
if (rval.eq.2) print*, ' Now run WIGFUN4b'
if (rval.eq.3) print*, ' Now run WIGFUN4c'
print*

```

c Format statements

```

905 format(2x,a50,2x,a23,2x,i8)
907 format(2x,a50,2x,i8,2x,f16.8)
908 format(2x,i8,2x,i8,2x,i8,2x,e16.8)
909 format(2x,a50,2x,f16.8)
end

```

c SUBROUTINES

```

include 'DATAOUT2.INCLUDE'
include 'PSEUDO.INCLUDE'
include 'REDUCE.INCLUDE'

```

```

c
c      [rossano.thesis]wigfun4a.for
c
c      Graham W. Rossano
c      Prof. J. Hamilton
c      Prof. Y.S. Shin
c
c      This program is the 4th of four to calculate and plot the
c      Wigner Distribution Function and variations of a signal
c
c      WIGFUN4a plots the distributions which were reduced to
c      64 x 32
c
c      All of the subroutines have been broken off for ease of modifying
c      and printing. Upon compiling, they are all included.
c
c      A description of the symbols used in this program and its
c      subroutines may be found in [rossano.thesis]symbols.include
c

```

```

real mtime,mfreq,dt,df,rwdf(64,32),ampl
character*23 datetime,tcode
character*25 inname,lfile
character*50 title,outname,titlehold
character*80 label
integer*4 status
integer dp,atvq,dimt,dimf,rnq,dp2,sizedp,n,mnq,
&i,j,rval,rdp,rdp2

```

c Start time of program

```

status = lib$date_time (datetime)
print*
type *, datetime
sizedp=512

```

```

c Get data from WIGFUN3
print*
print*, ' Program to calculate and plot'
print*, ' Wigner Distribution Function'
print*
print*, ' WIGFUN4a reduced to 64 x 32'
print*
open (unit=7,file='wig3.log',status='old')
rewind 7
read(7,908) dimt,dimf,dp,dt
read(7,905) outname,tcode,mnq
read(7,907) title,rval,ampl
close (unit=7)
print*, ' Wig3.log is loaded'
if (rval.eq.1) then
    rdp=32
else
    print*
    print*, ' You are using the wrong program'
    if (rval.eq.2) print*, ' Run WIGFUN4b'
    if (rval.eq.3) print*, ' Run WIGFUN4c'
    stop
endif
if (dp.gt.sizedp) then
    print*
    print*, ' max number of points allowed is'
    print*, sizedp
    print*, ' dp='
    print*, dp
    print*
    print*, ' Check your dimensions'
    stop
endif

c Calculations
dp2=dp*2
rdp2=rdp*2
mtime=dp*dt
df=1./(4.*mtime)
mfreq=2.*dp*df

c Convert mtime to msec for plotting
mtime=mtime*1000.
print*
print*, ' Do you want to plot the reduced wdf?'
print*, ' (1 for yes, 2 for no)'
read(5,*) atvq
if (atvq.eq.1) then
    open (unit=3,file='rwdf.out',status='old')
    rewind 3
    do 100 j=1,rdp
        read(3,928) n
        read(3,929)
    do 200 i=1,rdp2
        read(3,926) rwdf(i,j)

```

```

200     continue
        read(3,929)
        read(3,929)
100     continue
        close (unit=3)
        titlehold=title
        if (title.eq. 'Pseudo Wigner Distribution') then
            title='Reduced Wigner Distribution'
        endif
        if (title.eq. 'Pseudo Wigner-Ville Distribution') then
            title='Reduced Wigner-Ville Distribution'
        endif
        print*
        print*, 'rwdf is loaded'
        print*
        print*, 'Do you want to plot the rwdf split screen view?'
        print*, '          (1 for yes, 2 for no)'
        read(5,*) atvq
        if (atvq.eq.1) call plot3dsplit32 (rwdf,outname,mtime,
&mfreq,dp,tcode,title,mnq,ampl)
        print*
        print*, 'Do you want to plot the rwdf single view?'
        print*, '          (1 for yes, 2 for no)'
        read(5,*) atvq
        if (atvq.eq.1) call plot3d32 (rwdf,outname,
&mtime,mfreq,dp,tcode,title,mnq,ampl)
        print*
        print*, 'Do you want to plot the rwdf contours?'
        print*, '          (1 for yes, 2 for no)'
        read(5,*) atvq
        if (atvq.eq.1) call plotcon32 (rwdf,outname,
&mtime,mfreq,dp,tcode,title,mnq,ampl)
        title=titlehold
    endif
    print*
    print*, 'Do you want to plot the pseudo wdf?'
    print*, '          (1 for yes, 2 for no)'
    read(5,*) atvq
    if (atvq.eq.1) then
        open (unit=3,file='pwwdf.out',status='old')
        rewind 3
        do 300 j=1,rdp
            read(3,928) n
            read(3,929)
            do 400 i=1,rdp2
                read(3,926) rwdf(i,j)
400         continue
            read(3,929)
            read(3,929)
300     continue
        close (unit=3)
        print*
        print*, 'pwwdf is loaded'
        print*
        print*
        print*, 'Do you want to plot the pwwdf split screen view?'

```

```

        print*, '                (1 for yes, 2 for no)'
        read(5,*) atvq
        if (atvq.eq.1) call plot3dsplit32 (rwdf,outname,mtime,
&mfreq,dp,tcode,title,mnq,ampl)
        print*
        print*, ' Do you want to plot the pwdf single view?'
        print*, '                (1 for yes, 2 for no)'
        read(5,*) atvq
        if (atvq.eq.1) call plot3d32 (rwdf,outname,
&mtime,mfreq,dp,tcode,title,mnq,ampl)
        print*
        print*, ' Do you want to plot the pwdf contours?'
        print*, '                (1 for yes, 2 for no)'
        read(5,*) atvq
        if (atvq.eq.1) call plotcon32 (rwdf,outname,
&mtime,mfreq,dp,tcode,title,mnq,ampl)
        endif
        print*, ' Do you want to rename std00001.dat?'
        print*, '                (1 for yes, 2 for no)'
        read(5,*) rnq
        print*
        if (rnq.eq.1) then
            print*, ' Enter Laser plot file name'
            read(5,904) lfile
            LABEL(1:20)='RENAME STD00001.DAT '
            LABEL(21:45)=lfile
            CALL LIB$SPAWN(LABEL)
        endif
        print*
        print*, ' Program is complete'
        print*

```

c Format statements

```

904  format(a25)
905  format(2x,a50,2x,a23,2x,i8)
907  format(2x,a50,2x,i8,2x,f16.8)
908  format(2x,i8,2x,i8,2x,i8,2x,e16.8)
909  format(2x,a50)
926  format(2x,e16.8)
928  format(2x,i6)
929  format(2x)
end

```

c SUBROUTINES

```

include 'PLOT3D32. INCLUDE'
include 'PLOT3DSPLIT32. INCLUDE'
include 'PLOTCON32. INCLUDE'
include 'RANGE32. INCLUDE'

```

```

c      [rossano.thesis]wigfun4b.for
c
c      Graham W. Rossano
c      Prof. J. Hamilton
c      Prof. Y.S. Shin
c

```



```

c      This program is the 4th of four to calculate and plot the
c      Wigner Distribution Function and variations of a signal
c
c      WIGFUN4b plots the distributions which were reduced to
c      128 x 64
c
c      All of the subroutines have been broken off for ease of modifying
c      and printing. Upon compiling, they are all included.
c
c      A description of the symbols used in this program and its
c      subroutines may be found in [rossano.thesis]symbols.include
c

```

```

real mtime,mfreq,dt,df,rwdf(128,64),ampl
character*23 datetime,tcode
character*25 inname,lfile
character*50 title,outname,titlehold
character*80 label
integer*4 status
integer dp,atvq,dimt,dimf,rnq,dp2,sizedp,n,mnq,
&i,j,rval,rdp,rdp2

```

```

c Start time of program
  status = lib$date_time (datetime)
  print*
  type *, datetime
  sizedp=512

c Get data from WIGFUN3
  print*
  print*, ' Program to calculate and plot '
  print*, ' Wigner Distribution Function '
  print*
  print*, ' WIGFUN4b reduced to 128 x 64 '
  print*
  open (unit=7,file='wig3.log',status='old')
  rewind 7
  read(7,908) dimt,dimf,dp,dt
  read(7,905) outname,tcode,mnq
  read(7,907) title,rval,ampl
  close (unit=7)
  print*, ' Wig3.log is loaded '
  if (rval.eq.2) then
    rdp=64
  else
    print*
    print*, ' You are using the wrong program '
    if (rval.eq.1) print*, ' Run WIGFUN4a '
    if (rval.eq.3) print*, ' Run WIGFUN4c '
    stop
  endif
  if (dp.gt.sizedp) then
    print*
    print*, ' max number of points allowed is '
    print*, sizedp
    print*, ' dp= '

```

```

        print*,dp
        print*
        print*, ' Check your dimensions'
        stop
    endif

c Calculations
    dp2=dp*2
    rdp2=rdp*2
    mtime=dp*dt
    df=1./(4.*mtime)
    mfreq=2.*dp*df

c Convert mtime to msec for plotting
    mtime=mtime*1000.
    print*
    print*, ' Do you want to plot the reduced wdf?'
    print*, '          (1 for yes, 2 for no)'
    read(5,*) atvq
    if (atvq.eq.1) then
        open (unit=3,file='rwdf.out',status='old')
        rewind 3
        do 100 j=1,rdp
            read(3,928) n
            read(3,929)
            do 200 i=1,rdp2
                read(3,926) rwdf(i,j)
200            continue
                read(3,929)
                read(3,929)
100        continue
        close (unit=3)
        titlehold=title
        if (title.eq. 'Pseudo Wigner Distribution') then
            title='Reduced Wigner Distribution'
        endif
        if (title.eq. 'Pseudo Wigner-Ville Distribution') then
            title='Reduced Wigner-Ville Distribution'
        endif
        print*
        print*, ' rwdf is loaded'
        print*
        print*, ' Do you want to plot the rwdf split screen view?'
        print*, '          (1 for yes, 2 for no)'
        read(5,*) atvq
        if (atvq.eq.1) call plot3dsplit64 (rwdf,outname,mtime,
&mfreq,dp,tcode,title,mnq,ampl)
        print*
        print*, ' Do you want to plot the rwdf single view?'
        print*, '          (1 for yes, 2 for no)'
        read(5,*) atvq
        if (atvq.eq.1) call plot3d64 (rwdf,outname,
&mtime,mfreq,dp,tcode,title,mnq,ampl)
        print*
        print*, ' Do you want to plot the rwdf contours?'
        print*, '          (1 for yes, 2 for no)'

```

```

        read(5,*) atvq
        if (atvq.eq.1) call plotcon64 (rwdf,outname,
&mtime,mfreq,dp,tcode,title,mnq,ampl)
        title=titlehold
    endif
    print*
    print*,' Do you want to plot the pseudo wdf?'
    print*,'          (1 for yes, 2 for no)'
    read(5,*) atvq
    if (atvq.eq.1) then
        open (unit=3,file='pwdf.out',status='old')
        rewind 3
        do 300 j=1,rdp
            read(3,928) n
            read(3,929)
            do 400 i=1,rdp2
                read(3,926) rwdf(i,j)
400            continue
            read(3,929)
            read(3,929)
300        continue
        close (unit=3)
        print*
        print*,' pwdf is loaded'
        print*
        print*
        print*,' Do you want to plot the pwdf split screen view?'
        print*,'          (1 for yes, 2 for no)'
        read(5,*) atvq
        if (atvq.eq.1) call plot3dsplit64 (rwdf,outname,mtime,
&mfreq,dp,tcode,title,mnq,ampl)
        print*
        print*,' Do you want to plot the pwdf single view?'
        print*,'          (1 for yes, 2 for no)'
        read(5,*) atvq
        if (atvq.eq.1) call plot3d64 (rwdf,outname,
&mtime,mfreq,dp,tcode,title,mnq,ampl)
        print*
        print*,' Do you want to plot the pwdf contours?'
        print*,'          (1 for yes, 2 for no)'
        read(5,*) atvq
        if (atvq.eq.1) call plotcon64 (rwdf,outname,
&mtime,mfreq,dp,tcode,title,mnq,ampl)
    endif
    print*,' Do you want to rename std00001.dat?'
    print*,'          (1 for yes, 2 for no)'
    read(5,*) rnq
    print*
    if (rnq.eq.1) then
        print*,' Enter Laser plot file name'
        read(5,904) lfile
        LABEL(1:20)='RENAME STD00001.DAT '
        LABEL(21:45)=lfile
        CALL LIB$SPAWN(LABEL)
    endif
    print*

```

```

        print*, ' Program is complete'
        print*

c Format statements
904  format(a25)
905  format(2x,a50,2x,a23,2x,i8)
907  format(2x,a50,2x,i8,2x,f16.8)
908  format(2x,i8,2x,i8,2x,i8,2x,e16.8)
909  format(2x,a50)
926  format(2x,e16.8)
928  format(2x,i6)
929  format(2x)
      end

c SUBROUTINES

      include 'PLOT3D64. INCLUDE'
      include 'PLOT3DSPLIT64. INCLUDE'
      include 'PLOTCON64. INCLUDE'
      include 'RANGE64. INCLUDE'

c
c      [rossano.thesis]wigfun4c.for
c
c      Graham W. Rossano
c      Prof. J. Hamilton
c      Prof. Y.S. Shin
c
c      This program is the 4th of four to calculate and plot the
c      Wigner Distribution Function and variations of a signal
c
c      WIGFUN4c plots the distributions which were reduced to
c      256 x 128
c
c      All of the subroutines have been broken off for ease of modifying
c      and printing. Upon compiling, they are all included.
c
c      A description of the symbols used in this program and its
c      subroutines may be found in [rossano.thesis]symbols.include
c

      real mtime,mfreq,dt,df,rwdf(256,128),ampl
      character*23 datetime,tcode
      character*25 inname,lfile
      character*50 title,outname,titlehold
      character*80 label
      integer*4 status
      integer dp,atvq,dimt,dimf,rnq,dp2,sizedp,n,mnq,
      &i,j,rval,rdp,rdp2

c Start time of program
      status = lib$date_time (datetime)
      print*
      type *, datetime
      sizedp=512

c Get data from WIGFUN3

```

```

print*
print*, ' Program to calculate and plot'
print*, ' Wigner Distribution Function'
print*
print*, ' WIGFUN4c reduced to 256 x 128'
print*
open (unit=7,file='wig3.log',status='old')
rewind 7
read(7,908) dimt,dimf,dp,dt
read(7,905) outname,tcode,mnq
read(7,907) title,rval,ampl
close (unit=7)
print*, ' Wig3.log is loaded'
if (rval.eq.3) then
    rdp=128
else
    print*
    print*, ' You are using the wrong program'
    if (rval.eq.1) print*, ' Run WIGFUN4a'
    if (rval.eq.2) print*, ' Run WIGFUN4b'
    stop
endif
if (dp.gt.sizedp) then
    print*
    print*, ' max number of points allowed is'
    print*, sizedp
    print*, ' dp='
    print*, dp
    print*
    print*, ' Check your dimensions'
    stop
endif

c Calculations
dp2=dp*2
rdp2=rdp*2
mtime=dp*dt
df=1./(4.*mtime)
mfreq=2.*dp*df

c Convert mtime to msec for plotting
mtime=mtime*1000.
print*
print*, ' Do you want to plot the reduced wdf?'
print*, ' (1 for yes, 2 for no)'
read(5,*) atvq
if (atvq.eq.1) then
    open (unit=3,file='rwdf.out',status='old')
    rewind 3
    do 100 j=1,rdp
        read(3,928) n
        read(3,929)
        do 200 i=1,rdp2
            read(3,926) rwdf(i,j)
200        continue
        read(3,929)

```

```

        read(3,929)
100    continue
        close (unit=3)
        titlehold=title
        if (title.eq. 'Pseudo Wigner Distribution') then
            title='Reduced Wigner Distribution'
        endif
        if (title.eq. 'Pseudo Wigner-Ville Distribution') then
            title='Reduced Wigner-Ville Distribution'
        endif
        print*
        print*, 'rwdf is loaded'
        print*
        print*, 'Do you want to plot the rwdf split screen view?'
        print*, '          (1 for yes, 2 for no)'
        read(5,*) atvq
        if (atvq.eq.1) call plot3dsplit128 (rwdf,outname,mtime,
&mfreq,dp,tcode,title,mnq,ampl)
        print*
        print*, 'Do you want to plot the rwdf single view?'
        print*, '          (1 for yes, 2 for no)'
        read(5,*) atvq
        if (atvq.eq.1) call plot3d128 (rwdf,outname,
&mtime,mfreq,dp,tcode,title,mnq,ampl)
        print*
        print*, 'Do you want to plot the rwdf contours?'
        print*, '          (1 for yes, 2 for no)'
        read(5,*) atvq
        if (atvq.eq.1) call plotcon128 (rwdf,outname,
&mtime,mfreq,dp,tcode,title,mnq,ampl)
        title=titlehold
        endif
        print*
        print*, 'Do you want to plot the pseudo wdf?'
        print*, '          (1 for yes, 2 for no)'
        read(5,*) atvq
        if (atvq.eq.1) then
            open (unit=3,file='pwwdf.out',status='old')
            rewind 3
            do 300 j=1,rdp
                read(3,928) n
                read(3,929)
                do 400 i=1,rdp2
                    read(3,926) rwdf(i,j)
400                continue
                read(3,929)
                read(3,929)
300            continue
            close (unit=3)
            print*
            print*, 'pwwdf is loaded'
            print*
            print*
            print*, 'Do you want to plot the pwwdf split screen view?'
            print*, '          (1 for yes, 2 for no)'
            read(5,*) atvq

```

```

        if (atvq.eq.1) call plot3dsplit128 (rwdf,outname,mtime,
&mfreq,dp,tcode,title,mnq,ampl)
        print*
        print*, ' Do you want to plot the pwdf single view?'
        print*, '          (1 for yes, 2 for no)'
        read(5,*) atvq
        if (atvq.eq.1) call plot3d128 (rwdf,outname,
&mtime,mfreq,dp,tcode,title,mnq,ampl)
        print*
        print*, ' Do you want to plot the pwdf contours?'
        print*, '          (1 for yes, 2 for no)'
        read(5,*) atvq
        if (atvq.eq.1) call plotcon128 (rwdf,outname,
&mtime,mfreq,dp,tcode,title,mnq,ampl)
        endif
        print*, ' Do you want to rename std00001.dat?'
        print*, '          (1 for yes, 2 for no)'
        read(5,*) rnq
        print*
        if (rnq.eq.1) then
            print*, ' Enter Laser plot file name'
            read(5,904) lfile
            LABEL(1:20)='RENAME STD00001.DAT '
            LABEL(21:45)=lfile
            CALL LIB$SPAWN(LABEL)
        endif
        print*
        print*, ' Program is complete'
        print*

```

c Format statements

```

904  format(a25)
905  format(2x,a50,2x,a23,2x,i8)
907  format(2x,a50,2x,i8,2x,f16.8)
908  format(2x,i8,2x,i8,2x,i8,2x,e16.8)
909  format(2x,a50)
926  format(2x,e16.8)
928  format(2x,i6)
929  format(2x)
end

```

c SUBROUTINES

```

include 'PLOT3D128.INCLUDE'
include 'PLOT3DSPLIT128.INCLUDE'
include 'PLOTCON128.INCLUDE'
include 'RANGE128.INCLUDE'

```

C. ALPHANUMERIC LISTING OF SUBROUTINES

```

c
c      [rossano.thesis] amplify.include
c
c      This subroutine amplifies the signal ain
c
c
c      subroutine amplify (dimt,ain,dp,ampl)

```

```

integer dp,j,dimt
real ain(dimt),ampl

print*, '  What value would you like to amplify the signal by?'
print*, '          (Do not forget the decimal point)'
read(5,901) ampl
do 100 j=1,dp
    ain(j)=ampl*ain(j)
100 continue
return
901 format(f16.8)
end

c
c      [rossano.thesis] analytic. include
c
c      This subroutine converts a real signal into an analytic one
c

subroutine analytic (dimt,sr,s,dp)

integer dimt,dp,i,j
complex s(dimt)
real sr(dimt),dt,pi,sum,val,n,sumb,sval

pi=4.*atan(1.)

do 100 i=1,dp
    sum=0.
    do 200 j=1,dp
        sumb=0.
        if(i-j.eq.0) go to 200
        n=i-j
        val=pi*n/2.
        sval=sin(val)
        sumb=sr(j)*sval*sval/val
200    sum=sum+sumb
100 s(i)=cmplx(sr(i),sum)
return
end

c
c      [rossano.thesis] corr. include
c
c      This subroutine calculates the correlation
c

subroutine corr (dimt,dimf,s,j,dt,c,dp)

integer dimt,dimf,dp,dp2,i,j
complex s(dimt),c(dimf),dum
real coef,dt

dp2=dp*2
coef=2.*dt
do 100 i=1,dp+1
    if(j.ge.i) then

```



```

        dum=s(j-i+1)
      else
        dum=cplx(0.,0.)
      endif
      c(i)=coef*(s(j+i-1)*conjg(dum))
      if(i.ne.j.and.i.ne.dp+1) then
        c(dp2-i+2)=conjg(c(i))
      endif
100  continue
      return
      end

c
c      [rossano.thesis]datain.include
c
c      This subroutine reads the time signal data into an array
c
c  ASSUME input file in format: time  amplitude (2x,e16.8,5x,e16.8)
c  ASSUME end of file indication is a last entry of 9999.,9999.
c  Correct format may be obtained using [rossano.data]convert.for
c

      subroutine datain (dimt,inname,avgdelt,dp,ain,dt)

      integer j,i,dp,ss,dpnd,n,dimt
      real tin(2050),ain(dimt),amp,tim,mtime,mfreq,avgdelt,df,dt
      character*25 inname

      open (unit=4,file=inname,status='old')
      rewind 4

      mtime=dp*avgdelt
      df=1./(4.*mtime)
      mfreq=2.*dp*df

      print*,' Max time is'
      write(5,906) mtime
      print*,' Delta f is'
      write(5,906) df
      print*,' Max frequency is now'
      write(5,906) mfreq
      print*
      print*,' This can be changed by multiplying the delta t step size'
      print*,' if step size = 1 ; Max freq remains'
      write(5,906) mfreq

      print*,' if step size = 2 ; Max freq becomes      df=      mtime='
      mtime=dp*2.*avgdelt
      df=1./(4.*mtime)
      mfreq=2.*dp*df
      write(5,907) mfreq,df,mtime

      print*,' If step size = 5 ; Max freq becomes      df=      mtime='
      mtime=dp*5*avgdelt
      df=1./(4.*mtime)
      mfreq=2.*dp*df
      write(5,907) mfreq,df,mtime

```

```

print*, ' If step size = 10 ; Max freq becomes   df=           mtime='
mtime=dp*10*avgdelt
df=1./(4.*mtime)
mfreq=2.*dp*df
write(5,907) mfreq,df,mtime

print*, ' If step size = 16 ; Max freq becomes   df=           mtime='
mtime=dp*16*avgdelt
df=1./(4.*mtime)
mfreq=2.*dp*df
write(5,907) mfreq,df,mtime

print*
print*, ' Input desired step size'
read(*,*) ss
print*

do 111 j=1,dimt
    ain(j)=0.0
    tin(j)=0.0
111 continue

i=0
n=ss-1
dpend=dp*ss
do 200 j=1,dpend
    n=n+1
    read(4,902) tim,amp
    if (tim.eq.9999.) then
        if (amp.eq.9999.) go to 210
    endif
    if (n.eq.ss) then
        i=i+1
        ain(i)=amp
        tin(i)=tim
        n=0
    endif
200 continue
go to 212

c This pads zeros if the signal is too short
210 j=i+1
print*, ' Padding with zeros since not enough data points'
print*
do 211 j=j,dp
    ain(j)=0.0
    tin(j)=j*ss*avgdelt
211 continue

212 close (unit=4)
mtime=tin(dp)
dt=mtime/(dp-1)

call dtcalc (dimt,dp,inname,tin,avgdelt)

```

```

print*, ' Average dt='
write(5,906) avgdelt
print*, ' Final dt='
write(5,906) dt
print*
if (avgdelt.ne.dt) then
  print*, ' Average dt does not equal Final dt'
  print*, ' This is probably because your time record'
  print*, ' did not start at zero.'
  print*, ' This can be fixed by letting dt=average dt'
  print*, ' Is this ok?'
  print*, '          (1 for yes, 2 for no)'
  read(5,*) atvq
  if (atvq.eq.2) stop
  dt=avgdelt
endif
return

c Format statements
902 format(2x,e16.8,5x,e16.8)
906 format(f16.8)
907 format(20x,f16.8,f16.8,f16.8)
end

c
c   [rossano.thesis]dataout.include
c
c   This subroutine prints the WDF array to a file
c

subroutine dataout (dp)

integer j,i,dp,dp2
real wdf(1100,550)
common /wdfc/ wdf

open (unit=7,file='wdf.unf',status='new',
&form='unformatted')
  print*, ' Writing to wdf.unf'

  dp2=dp*2
  do 100 j=1,dp
    do 200 i=1,dp2
      write(7) wdf(i,j)
200   continue
100  continue
  close (unit=7)
  return
end

c
c   [rossano.thesis]dataout2.include
c
c   This subroutine prints the RWDF and PWDF arrays to files
c

subroutine dataout2 (rwdf,n,dp,nn,mm)

```

```

integer j,i,n,dp,dp2,nn,mm
real rwdf (256,128)

if (n.eq.2) then
  open (unit=7,file='pwdf.out',status='new')
  print*, ' Writing to pwdf.out'
endif
if (n.eq.3) then
  open (unit=7,file='rwdf.out',status='new')
  print*, ' Writing to rwdf.out'
endif
dp2=dp*2
do 100 j=1,dp/mm
  write(7,908) j
  write(7,909)
  do 200 i=1,dp2/nn
    write(7,906) rwdf(i,j)
200   continue
    write(7,909)
    write(7,909)
100  continue
close (unit=7)
return
906 format (2x,e16.8)
908 format (2x,i6)
909 format (2x)
end

c
c   [rossano.thesis] dtcalc. include
c
c   This subroutine calculates delta t from an input file (j=1) or
c   an array (j=dp)
c

subroutine dtcalc (dimt,j,inname,tin,avgdelt)

integer i,j,n,dimt
real tin(dimt),ain(1025),sum,delt(1025),avgdelt
character*25 inname

n=0
sum=0.

if (j.ne.1) then
  n=j
  go to 300
endif

open (unit=4,file=inname,status='old')
rewind 4

do 100 i=1,1025
  read(4,904) tin(i),ain(i)
  if (tin(i).eq.9999.) then
    if (ain(i).eq.9999.) go to 200
  endif

```

```

      n=n+1
100  continue
200  close (unit=4)

300  do 400 i=1,n-1
      delt(i)=tin(i+1)-tin(i)
      sum=sum+delt(i)
400  continue

      avgdelt=sum/(n-1)
      return
904  format (2x,e16.8,5x,e16.8)
      end

c
c      [rossano.thesis] fft. include
c
c      This subroutine calculates the Fast Fourier Transform
c      (FFT if inv=0, Inverse FFT if inv=1)
c

      subroutine fft (dimf,c,dp2,inv)

      integer dimf,inv,dp,dp2,i,j,n,val,ii,k
      complex c(dimf),dum,dum2,dum3
      real pi,const,coef

      pi=4.*atan(1.)
      const=dp2
      val=alog(const)/alog(2.)+.1
      dp=dp2/2
      j=1
      do 40 i=1,dp2-1
        if (i.ge.j) go to 10
        dum3=c(j)
        c(j)=c(i)
        c(i)=dum3
10      k=dp
20      if (k.ge.j) go to 30
        j=j-k
        k=k/2
        go to 20
30      j=j+k
40      continue
      do 70 n=1,val
        coef=2**n
        coef=coef/2
        dum2=cplx(1.,0.)
        dum=cplx(cos(pi/coef),-sin(pi/coef))
        if (inv.ne.0) dum=conjg(dum)
        do 60 j=1,coef
          do 50 i=j,dp2,2*coef
            ii=i+coef
            dum3=c(ii)*dum2
            c(ii)=c(i)-dum3
            c(i)=c(i)+dum3
50          continue

```

```

        dum2=dum2*dum
60    continue
70    continue
    if (inv.eq.0) return
    do 80 i=1,dp2
        c(i)=c(i)/cplx(const,0.)
80    continue
    return
end

c
c    [rossano.thesis]hamming.include
c
c    This subroutine applies a hamming window to the signal ain
c

subroutine hamming (dimt,ain,dp,dt)

integer j,dp,dimt
real ain(dimt),dt,mtime,pi,const,delta,n

pi=4.0*atan(1.0)
mtime=dp*dt
delta=0.1*mtime
const=pi/delta

do 100 j=1,dp
    n=(j-1)*dt
    if (n.le.delta) ain(j)=ain(j)*(.5*(1.-cos(const*n)))
    if (n.ge.mtime-delta) then
        ain(j)=ain(j)*(.5*(1.-cos(const*(mtime-n))))
    endif
100 continue
return
end

c
c    [rossano.thesis]hanning.include
c
c    This subroutine applies a hanning window to signal ain
c

subroutine hanning (dimt,ain,dp,dt)

integer j,dp,dimt
real ain(dimt),dt,mtime,pi2,windw,const

pi2=2.*4.0*atan(1.0)
mtime=dp*dt
const=(pi2*dt)/mtime
do 100 j=0,dp-1
    windw=0.5*(1-cos(const*j))
    ain(j+1)=ain(j+1)*windw
100 continue
return
end

c
c    [rossano.thesis]maxamp.include

```

```

c
c      This subroutine finds the maximum amplitude of array y
c
      subroutine maxamp (y,dp,zmax)
c
      integer i,dp
      real y(dp),zmax,zhold
c
      zmax=0.0
      do 200 i=1,dp
        zhold=y(i)
        if (zhold.gt.zmax) then
          zmax=zhold
        endif
c
200    continue
      return
      end
c
c      [rossano.thesis]mean.include
c
c      This subroutine calculates the mean value in the signal ain
c
      subroutine mean (dimt,ain,dp,meanv)
c
      integer dp,i,dimt
      real ain(dimt),sum,meanv
c
      sum=0.0
      do 100 i=1,dp
        sum=sum+ain(i)
c
100    continue
      meanv=sum/dp
      return
      end
c
c      [rossano.thesis]meanr.include
c
c      This subroutine removes the mean value from the signal ain
c
      subroutine meanr (dimt,ain,dp,meanv)
c
      integer dp,i,dimt
      real ain(dimt),meanv
c
      do 100 i=1,dp
        ain(i)=ain(i)-meanv
c
100    continue
      return
      end
c
c      [rossano.thesis]minamp.include
c

```

```

c      This subroutine finds the minimum amplitude of array y
c
      subroutine minamp (y,dp,zmin)

      integer i,dp
      real y(dp),zmin,zhold

      zmin=0.0
      do 200 i=1,dp
        zhold=y(i)
        if (zhold.lt.zmin) then
          zmin=zhold
        endif
200    continue
      return
      end

c
c      [rossano.thesis]plot2d.include
c
c      This subroutine uses disspla 11.0 to plot a 2 Dimension plot
c
c The data input array is y, the x axis assumes a constant dt spacing
c

      subroutine plot2d (y,dt,dp,title,outname)

      real dt,mtime,zmax,zmin
      character*23 datetime
      character*50 title,outname
      integer*4 status
      integer iplot_val,dp,mesh,j
      include 'size.include'

      call maxamp (y,dp,zmax)
      call minamp (y,dp,zmin)

      mtime=dp*dt
      do 100 j=1,dp
        x(j)=(j-1)*dt
100    continue

      print*
      print*, ' Enter limits for plotting (do not forget the decimal)'
      print*, '          YMIN          YMAX'
      write(5,*) zmin,zmax
      read(5,*) zmin,zmax
      print*

      mesh=1
      call reset ('all')
      write(5,*) ' Do you want to view the plot on the screen or get
& a hardcopy?'
      write(5,*) '          (1 for view, 2 for hardcopy)'
      print*
      read(5,*) iplot_val

```



```

print*
if (iplot_val .eq. 1) then
  print*, ' When finished viewing hit return key'
  call pgpx
endif
if (iplot_val .eq. 2) then
  print*, ' Please be patient'
  print*, ' This will take several minutes.'
  call LN03I
endif
call swissm
call hws hd
call chrpat (16)
call height (.325)
call physor (1.0,0.625)
call area2d (6.0,8.5)
call alnmes (.5,0.)
call messag (title,100,3.2,9.9)
call messag (outname,100,3.2,9.5)
call reset ('alnmes')
call blsur
call height (0.200)
call xname ('Time (sec)',10)
call yname ('Amplitude',9)
call cross
call xintax
call yintax
call graf (0.,mtime/4.,mtime,zmin,(zmax-zmin)/4.,zmax)
print*, ' Axes are complete'
call reset ('hws hd')
call curve (x,y,dp,0)
call height (.150)
call alnmes (0.0,1.0)
status = lib$date_time (datetime)
call messag (datetime,23,-0.8,-0.4)
call reset ('alnmes')
510 print*, ' Surface is complete'
call end3gr (0)
call endpl (0)
print*, ' Plotting complete'
print*
return
end

c
c      [rossano.thesis]plot2d2.include
c
c      This subroutine uses disspla 11.0 to plot 2 2 Dimension curves
c      on one plot
c
c The data input arrays are x and y, the x axis assumes a constant dt
c spacing
c

subroutine plot2d2 (x,y,dt,dp,title,outname,mnq)

real dt,mtime,zmax,zmin,xmax,xmin,ymax,ymin

```

```

character*23 datetime
character*50 title,outname,meani
integer*4 status
integer iplot_val,dp,mesh,j,mnq
include 'size.include'

mtime=dp*dt
mesh=1

if (mnq.eq.1) then
  meani='Mean value removed'
endif
if (mnq.eq.2) then
  meani='Mean value not removed'
endif

call maxamp (x,dp,zmax)
call minamp (x,dp,zmin)
xmax=zmax
xmin=zmin
call maxamp (y,dp,zmax)
call minamp (y,dp,zmin)
ymax=zmax
ymin=zmin
if (xmax.gt.ymax) zmax=xmax
if (xmin.lt.ymin) zmin=xmin

print*
print*, ' Enter limits for plotting (do not forget the decimal)'
print*, '      YMIN      YMAX'
write(5,*) zmin,zmax
read(5,*) zmin,zmax
print*

do 100 j=1,dp
  t(j)=(j-1)*dt
100 continue

call reset ('all')
write(5,*) ' Do you want to view the plot on the screen or get
& a hardcopy?'
write(5,*) '      (1 for view, 2 for hardcopy)'
print*
read(5,*) iplot_val
print*
if (iplot_val .eq. 1) then
  print*, ' When finished viewing hit return key'
  call pgpx
endif
if (iplot_val .eq. 2) then
  print*, ' Please be patient'
  print*, ' This will take several minutes.'
  call LN03I
endif
call swissm
call hwshd

```

```

call chrpat (16)
call height (.325)
call physor (1.0,0.625)
call area2d (6.0,8.5)
call alnmes (.5,0.)
call messag (title,100,3.2,9.9)
call messag (outname,100,3.2,9.5)
call reset ('alnmes')
call blsur
call height (0.200)
call xname ('Time (sec)',10)
call yname ('Amplitude',9)
call cross
call xintax
call yintax
call graf (0.,mtime/4.,mtime,zmin,(zmax-zmin)/4.,zmax)
print*, ' Axes are complete'
call reset ('hwsht')
call curve (t,x,dp,0)
call dot
call curve (t,y,dp,0)
call height (.150)
call alnmes (0.0,1.0)
status = lib$date_time (datetime)
call messag (datetime,23,-0.8,-0.1)
call messag (meani,50,-0.8,-0.4)
call messag ('Solid line=REAL',15,3.0,-0.1)
call messag ('Dotted line=IMAGINARY',21,3.0,-0.4)
call reset ('alnmes')
510 print*, ' Surface is complete'
call end3gr (0)
call endpl (0)
print*, ' Plotting complete'
print*
return
end

c
c   [rossano.thesis]plot3d32.include
c
c   This subroutine uses disspla 11.0 to plot a single 3 dimension
c   graph of rwdf(64,32)
c

subroutine plot3d32 (rwdf,outname,mtime,mfreq,dp,
&tcode,title,mnq,ampl)

real mtime,mfreq,zmax,zmin,ampl
character*23,datetime,tcode,dpnum,amplif
character*50,title,outname,meani
integer*4 status
integer iplot_val,dp,mesh,rdp2,rdp,mnq
real rwdf(64,32)

rdp=32
if (dp.eq.128) then
  dpnum='128 data points'

```

```

endif
if (dp.eq.256) then
    dpnum='256 data points'
endif
if (dp.eq.512) then
    dpnum='512 data points'
endif
if (dp.eq.1024) then
    dpnum='1024 data points'
endif
if (mnq.eq.1) then
    meani='Mean value removed'
endif
if (mnq.eq.2) then
    meani='Mean value not removed'
endif
amplif='Time amplified'
if (ampl.eq.0.0) amplif='Time amplified by 0.0'
if (ampl.eq.1.0) amplif='Time amplified by 1.0'
if (ampl.eq.2.0) amplif='Time amplified by 2.0'
if (ampl.eq.3.0) amplif='Time amplified by 3.0'
if (ampl.eq.4.0) amplif='Time amplified by 4.0'
if (ampl.eq.5.0) amplif='Time amplified by 5.0'
call range32 (rwdf,rdp,zmax,zmin)
print*
print*, ' Enter limits for plotting (do not forget decimal)'
print*, '      ZMIN      ZMAX'
write(5,*) zmin,zmax
read(5,*) zmin,zmax
print*
mesh=1
rdp2=rdp*2
call reset ('all')
write(5,*) ' Do you want to view the plot on the screen or get
& a hardcopy?'
write(5,*) '      (1 for view, 2 for hardcopy)'
print*
read(5,*) iplot_val
print*
if (iplot_val .eq. 1) then
    print*, ' When finished viewing hit return key'
    call pgpx
endif
if (iplot_val .eq. 2) then
    print*, ' Please be patient'
    print*, ' This will take several minutes.'
    call LN03I
endif
call swissm
call hwshd
call chrpat (16)
call height (.325)
call physor (.75,.625)
call area2d (7.5,9.75)
call alnmes (.5,0.)
call messag (title,100,7.5/2.,9.25)

```

```

call messag (outname,100,7.5/2.,8.75)
call reset ('alnmes')
call blsur
call volm3d (8.,8.,9.)
call x3name ('Frequency (Hz)',15)
call y3name ('Time (msec)',11)
call z3name ('Amplitude',9)
call vuangl (-60.,30.,30.)
call xintax
call yintax
call zintax
call graf3d (0.,mfreq/4.,mfreq,0.,mtime/2.,mtime,zmin,
&(zmax-zmin)/3.,zmax)
print*, ' Axes are complete'
call reset ('hwshd')
call surmat (rwdf,mesh,rdp2,mesh,rdp,0)
call reset ('alnmes')
call height (0.150)
call alnmes (0.0,1.0)
call messag (meani,50,5.0,0.0)
call messag (amplif,23,5.0,-0.2)
call messag (tcode,23,5.0,-0.4)
status = lib$date_time (datetime)
call messag (datetime,23,0.0,0.2)
call messag (dpnum,23,0.0,0.0)
call messag ('Reduced to 64 x 32',23,0.0,-0.2)
call messag ('Smoothed 10 x 10',23,0.0,-0.4)
call reset ('alnmes')
print*, ' Surface is complete'
510 call end3gr (0)
call endpl (0)
print*, ' Plotting complete'
print*
return
end

c
c   [rossano.thesis]plot3d64.include
c
c   This subroutine uses disspla 11.0 to plot a single 3 dimension
c   graph of rwdf(128,64)
c

subroutine plot3d64 (rwdf,outname,mtime,mfreq,dp,
&tcode,title,mnq,ampl)

real mtime,mfreq,zmax,zmin,ampl
character*23,datetime,tcode,dpnum,amplif
character*50,title,outname,meani
integer*4 status
integer iplot_val,dp,mesh,rdp2,rdp,mnq
real rwdf(128,64)

rdp=64
if (dp.eq.128) then
  dpnum='128 data points'
endif

```

```

if (dp.eq.256) then
  dpnum='256 data points'
endif
if (dp.eq.512) then
  dpnum='512 data points'
endif
if (dp.eq.1024) then
  dpnum='1024 data points'
endif
if (mnq.eq.1) then
  meani='Mean value removed'
endif
if (mnq.eq.2) then
  meani='Mean value not removed'
endif
amplif='Time amplified'
if (ampl.eq.0.0) amplif='Time amplified by 0.0'
if (ampl.eq.1.0) amplif='Time amplified by 1.0'
if (ampl.eq.2.0) amplif='Time amplified by 2.0'
if (ampl.eq.3.0) amplif='Time amplified by 3.0'
if (ampl.eq.4.0) amplif='Time amplified by 4.0'
if (ampl.eq.5.0) amplif='Time amplified by 5.0'
call range64 (rwdf,rdp,zmax,zmin)
print*
print*, ' Enter limits for plotting (do not forget decimal)'
print*, '      ZMIN      ZMAX'
write(5,*) zmin,zmax
read(5,*) zmin,zmax
print*
mesh=1
rdp2=rdp*2
call reset ('all')
write(5,*) ' Do you want to view the plot on the screen or get
& a hardcopy?'
write(5,*) '      (1 for view, 2 for hardcopy)'
print*
read(5,*) iplot_val
print*
if (iplot_val .eq. 1) then
  print*, ' When finished viewing hit return key'
  call pgpx
endif
if (iplot_val .eq. 2) then
  print*, ' Please be patient'
  print*, ' This will take several minutes.'
  call LN03I
endif
call swissm
call hws hd
call chrpat (16)
call height (.325)
call physor (.75,.625)
call area2d (7.5,9.75)
call alnmes (.5,0.)
call messag (title,100,7.5/2.,9.25)
call messag (outname,100,7.5/2.,8.75)

```

```

call reset ('alnmes')
call blsur
call volm3d (8.,8.,9.)
call x3name ('Frequency (Hz)',15)
call y3name ('Time (msec)',11)
call z3name ('Amplitude',9)
call vuangl (-60.,30.,30.)
call xintax
call yintax
call zintax
call graf3d (0.,mfreq/4.,mfreq,0.,mtime/2.,mtime,zmin,
&(zmax-zmin)/3.,zmax)
print*, ' Axes are complete'
call reset ('hwsd')
call surmat (rwdf,mesh,rdp2,mesh,rdp,0)
call reset ('alnmes')
call height (0.150)
call alnmes (0.0,1.0)
call messag (meani,50,5.0,0.0)
call messag (amplif,23,5.0,-0.2)
call messag (tcode,23,5.0,-0.4)
status = lib$date_time (datetime)
call messag (datetime,23,0.0,0.2)
call messag (dpnum,23,0.0,0.0)
call messag ('Reduced to 128 x 64',23,0.0,-0.2)
call messag ('Smoothed 10 x 10',23,0.0,-0.4)
call reset ('alnmes')
510 print*, ' Surface is complete'
call end3gr (0)
call endpl (0)
print*, ' Plotting complete'
print*
return
end

c
c      [rossano.thesis]plot3d128.include
c
c      This subroutine uses disspla 11.0 to plot a single 3 dimension
c      graph of rwdf(256,128)
c

subroutine plot3d128 (rwdf,outname,mtime,mfreq,dp,
&tcode,title,mnq,ampl)

real mtime,mfreq,zmax,zmin,ampl
character*23,datetime,tcode,dpnum,amplif
character*50,title,outname,meani
integer*4 status
integer iplot_val,dp,mesh,rdp2,rdp,mnq
real rwdf(256,128)

rdp=128
if (dp.eq.128) then
    dpnum='128 data points'
endif
if (dp.eq.256) then

```

```

    dpnum='256 data points'
endif
if (dp.eq.512) then
    dpnum='512 data points'
endif
if (dp.eq.1024) then
    dpnum='1024 data points'
endif
if (mnq.eq.1) then
    meani='Mean value removed'
endif
if (mnq.eq.2) then
    meani='Mean value not removed'
endif
amplif='Time amplified'
if (ampl.eq.0.0) amplif='Time amplified by 0.0'
if (ampl.eq.1.0) amplif='Time amplified by 1.0'
if (ampl.eq.2.0) amplif='Time amplified by 2.0'
if (ampl.eq.3.0) amplif='Time amplified by 3.0'
if (ampl.eq.4.0) amplif='Time amplified by 4.0'
if (ampl.eq.5.0) amplif='Time amplified by 5.0'
call rangel28 (rwdf,rdp,zmax,zmin)
print*
print*, ' Enter limits for plotting (do not forget decimal)'
print*, '      ZMIN      ZMAX'
write(5,*) zmin,zmax
read(5,*) zmin,zmax
print*
mesh=1
rdp2=rdp*2
call reset ('all')
write(5,*) ' Do you want to view the plot on the screen or get
& a hardcopy?'
write(5,*) '      (1 for view, 2 for hardcopy)'
print*
read(5,*) iplot_val
print*
if (iplot_val.eq. 1) then
    print*, ' When finished viewing hit return key'
    call pgpx
endif
if (iplot_val.eq. 2) then
    print*, ' Please be patient'
    print*, ' This will take several minutes.'
    call LN03I
endif
call swissm
call hws hd
call chrpat (16)
call height (.325)
call physor (.75,.625)
call area2d (7.5,9.75)
call alnmes (.5,0.)
call messag (title,100,7.5/2.,9.25)
call messag (outname,100,7.5/2.,8.75)
call reset ('alnmes')

```



```

call blsur
call volm3d (8.,8.,9.)
call x3name ('Frequency (Hz)',15)
call y3name ('Time (msec)',11)
call z3name ('Amplitude',9)
call vuangl (-60.,30.,30.)
call xintax
call yintax
call zintax
call graf3d (0.,mfreq/4.,mfreq,0.,mtime/2.,mtime,zmin,
&(zmax-zmin)/3.,zmax)
print*, ' Axes are complete'
call reset ('hwsht')
call surmat (rwdf,mesh,rdp2,mesh,rdp,0)
call reset ('alnmes')
call height (0.150)
call alnmes (0.0,1.0)
call messag (mean1,50,5.0,0.0)
call messag (amplif,23,5.0,-0.2)
call messag (tcode,23,5.0,-0.4)
status = lib$date_time (datetime)
call messag (datetime,23,0.0,0.2)
call messag (dpnum,23,0.0,0.0)
call messag ('Reduced to 256 x 128',23,0.0,-0.2)
call messag ('Smoothed 10 x 10',23,0.0,-0.4)
call reset ('alnmes')
510 print*, ' Surface is complete'
call end3gr (0)
call endpl (0)
print*, ' Plotting complete'
print*
return
end

c
c      [rossano.thesis]plot3dsplit32.include
c
c      This subroutine uses disspla 11.0 to plot a series of
c      3 dimensional graphs of rwdf(64,32)
c

      subroutine plot3dsplit32 (rwdf,outname,mtime,mfreq,dp,tcode,
&title,mnq,ampl)

      real mtime,mfreq,zmax,zmin,rwdf(64,32),ampl
      character*23,datetime,tcode,dpnum,amplif
      character*50,title,outname,mean1
      integer*4 status
      integer iplot_val,dp,mesh,rdp2,rdp,mnq

      rdp=32
      if (dp.eq.128) then
         dpnum='128 data points'
      endif
      if (dp.eq.256) then
         dpnum='256 data points'
      endif

```

```

if (dp.eq.512) then
  dpnum='512 data points'
endif
if (dp.eq.1024) then
  dpnum='1024 data points'
endif
if (mnq.eq.1) then
  meani='Mean value removed'
endif
if (mnq.eq.2) then
  meani='Mean value not removed'
endif
amplif='Time amplified'
if (ampl.eq.0.0) amplif='Time amplified by 0.0'
if (ampl.eq.1.0) amplif='Time amplified by 1.0'
if (ampl.eq.2.0) amplif='Time amplified by 2.0'
if (ampl.eq.3.0) amplif='Time amplified by 3.0'
if (ampl.eq.4.0) amplif='Time amplified by 4.0'
if (ampl.eq.5.0) amplif='Time amplified by 5.0'
call range32 (rwdf,rdp,zmax,zmin)
print*, 'Enter limits for plotting (do not forget decimal)'
print*, '      ZMIN      ZMAX'
write(5,*) zmin,zmax
read(5,*) zmin,zmax
print*
mesh=1
rdp2=rdp*2
call reset ('all')
write(5,*) 'Do you want to view the plot on the screen or get
& a hardcopy?'
write(5,*) '      (1 for view, 2 for hardcopy)'
print*
read(5,*) iplot_val
print*
if (iplot_val.eq. 1) then
  print*, 'When finished viewing hit return key'
  call pgpx
endif
if (iplot_val.eq. 2) then
  print*, 'Please be patient'
  print*, 'This will take several minutes.'
  call LN03I
endif
call swissm
call hws hd
call chrpat (16)

```

c LABELS

```

call height (0.200)
call physor (0.5,0.625)
call area2d (7.5,9.75)
call alnmes (0.5,0.0)
call messag (title,100,7.5/2.,9.75)
call messag (outname,100,7.5/2.,9.5)
call reset ('alnmes')
call height (0.150)

```

```

call alnmes (0.0,1.0)
call messag (meani,50,5.0,0.0)
call messag (amplif,23,5.0,-0.2)
call messag (tcode,23,5.0,-0.4)
status = lib$date_time (datetime)
call messag (datetime,23,0.0,0.2)
call messag (dpnum,23,0.0,0.0)
call messag ('Reduced to 64 x 32',23,0.0,-0.2)
call messag ('Smoothed 10 x 10',23,0.0,-0.4)
call reset ('alnmes')
call endgr (1)
print*, 'Labels are complete'

```

c FIRST SUBPLOT

```

call height (.150)
call physor (.5,5.5)
call area2d (3.5,4.875)
call blsur
call volm3d (8.,8.,9.)
call x3name ('Frequency (Hz)',15)
call y3name ('Time (msec)',11)
call z3name ('Amplitude',9)
call vuangl (-50.,0.,30.)
call xintax
call yintax
call zintax
call graf3d (0.,mfreq/4.,mfreq,0.,mtime/2.,mtime,zmin,
&(zmax-zmin)/3.,zmax)
print*, 'First subplot axes are complete'
call reset ('hwshd')
call surmat (rwdf,mesh,rdp2,mesh,rdp,0)
call endgr (2)
print*, 'First subplot surface is complete'

```

c SECOND SUBPLOT

```

call height (.150)
call physor (4.25,5.5)
call area2d (3.5,4.5)
call blsur
call volm3d (8.,8.,9.)
call x3name ('',1)
call y3name ('Time (msec)',11)
call z3name ('Amplitude',9)
call vuangl (0.,0.,30.)
call yintax
call zintax
call xnonum
call graf3d (0.,mfreq/4.,mfreq,0.,mtime/2.,mtime,zmin,
&(zmax-zmin)/3.,zmax)
print*, 'Second subplot axes are complete'
call reset ('hwshd')
call surmat (rwdf,mesh,rdp2,mesh,rdp,0)
call endgr (3)
print*, 'Second subplot surface is complete'

```

c THIRD SUBPLOT

```

call reset ('xnonum')
call height (.150)
call physor (.5,.625)
call area2d (3.5,4.875)
call blsur
call volm3d (8.,8.,9.)
call x3name ('Frequency (Hz)',15)
call y3name ('',1)
call z3name ('Amplitude',9)
call vuangl (-90.,0.,30.)
call xintax
call zintax
call ynonum
call graf3d (0.,mfreq/4.,mfreq,0.,mtime/2.,mtime,zmin,
&(zmax-zmin)/3.,zmax)
print*, ' Third subplot axes are complete'
call reset ('hwshd')
call surmat (rwdf,mesh,rdp2,mesh,rdp,0)
call endgr (4)
print*, ' Third subplot surface is complete'

```

C FOURTH SUBPLOT

```

call reset ('ynonum')
call height (.125)
call physor (4.75,1.5)
call area2d (2.75,3.0)
call blsur
call xname ('Frequency (Hz)',15)
call yname ('Time (msec)',11)
call xintax
call yintax
call graf (0.,mfreq/4.,mfreq,0.,mtime/2.,mtime)
print*, ' Fourth subplot axes are complete'
call messag ('Contour at zmax/3',17,0.5,3.3)
call reset ('hwshd')
call conbgn
call zrange (zmax/3.,zmax/3.)
call conmak (rwdf,rdp2,rdp,1)
call conend
call conlin (1,'solid','nolabels',1,10)
call contur (1,'nolabels','draw')
call endgr (5)
print*, ' Fourth subplot surface is complete'
call endpl (0)
print*, ' Plotting complete'
print*
return
end

```

```

c
c      [rossano.thesis]plot3dsplit64.include
c
c      This subroutine uses disspla 11.0 to plot a series of
c      3 dimensional graphs of rwdf(128,64)
c

```

```

subroutine plot3dsplit64 (rwdf,outname,mtime,mfreq,dp,tcode,

```

```

&title,mnq,ampl)

real mtime,mfreq,zmax,zmin,rwdf(128,64),ampl
character*23,datetime,tcode,dpnum,amplif
character*50,title,outname,meani
integer*4 status
integer iplot_val,dp,mesh,rdp2,rdp,mnq

rdp=64
if (dp.eq.128) then
    dpnum='128 data points'
endif
if (dp.eq.256) then
    dpnum='256 data points'
endif
if (dp.eq.512) then
    dpnum='512 data points'
endif
if (dp.eq.1024) then
    dpnum='1024 data points'
endif
if (mnq.eq.1) then
    meani='Mean value removed'
endif
if (mnq.eq.2) then
    meani='Mean value not removed'
endif
amplif='Time amplified'
if (ampl.eq.0.0) amplif='Time amplified by 0.0'
if (ampl.eq.1.0) amplif='Time amplified by 1.0'
if (ampl.eq.2.0) amplif='Time amplified by 2.0'
if (ampl.eq.3.0) amplif='Time amplified by 3.0'
if (ampl.eq.4.0) amplif='Time amplified by 4.0'
if (ampl.eq.5.0) amplif='Time amplified by 5.0'
call range64 (rwdf,rdp,zmax,zmin)
print*,' Enter limits for plotting (do not forget decimal)'
print*,'          ZMIN          ZMAX'
write(5,*) zmin,zmax
read(5,*) zmin,zmax
print*
mesh=1
rdp2=rdp*2
call reset ('all')
write(5,*) ' Do you want to view the plot on the screen or get
& a hardcopy?'
write(5,*) '          (1 for view, 2 for hardcopy)'
print*
read(5,*) iplot_val
print*
if (iplot_val .eq. 1) then
    print*,' When finished viewing hit return key'
    call pgpx
endif
if (iplot_val .eq. 2) then
    print*,' Please be patient'
    print*,' This will take several minutes.'

```

```

        call LN03I
    endif
    call swissm
    call hwshd
    call chrpat (16)

```

c LABELS

```

    call height (0.200)
    call physor (0.5,0.625)
    call area2d (7.5,9.75)
    call alnmes (0.5,0.0)
    call messag (title,100,7.5/2.,9.75)
    call messag (outname,100,7.5/2.,9.5)
    call reset ('alnmes')
    call height (0.150)
    call alnmes (0.0,1.0)
    call messag (meani,50,5.0,0.0)
    call messag (amplif,23,5.0,-0.2)
    call messag (tcode,23,5.0,-0.4)
    status = lib$date_time (datetime)
    call messag (datetime,23,0.0,0.2)
    call messag (dpnum,23,0.0,0.0)
    call messag ('Reduced to 128 x 64',23,0.0,-0.2)
    call messag ('Smoothed 10 x 10',23,0.0,-0.4)
    call reset ('alnmes')
    call endgr (1)
    print*, 'Labels are complete'

```

c FIRST SUBPLOT

```

    call height (.150)
    call physor (.5,5.5)
    call area2d (3.5,4.875)
    call blsur
    call volm3d (8.,8.,9.)
    call x3name ('Frequency (Hz)',15)
    call y3name ('Time (msec)',11)
    call z3name ('Amplitude',9)
    call vuangl (-50.,0.,30.)
    call xintax
    call yintax
    call zintax
    call graf3d (0.,mfreq/4.,mfreq,0.,mtime/2.,mtime,zmin,
&(zmax-zmin)/3.,zmax)
    print*, 'First subplot axes are complete'
    call reset ('hwshd')
    call surmat (rwdf,mesh,rdp2,mesh,rdp,0)
    call endgr (2)
    print*, 'First subplot surface is complete'

```

c SECOND SUBPLOT

```

    call height (.150)
    call physor (4.25,5.5)
    call area2d (3.5,4.5)
    call blsur
    call volm3d (8.,8.,9.)
    call x3name ('',1)

```

```

call y3name ('Time (msec)',11)
call z3name ('Amplitude',9)
call vuangl (0.,0.,30.)
call yintax
call zintax
call xnonum
call graf3d (0.,mfreq/4.,mfreq,0.,mtime/2.,mtime,zmin,
&(zmax-zmin)/3.,zmax)
print*, ' Second subplot axes are complete'
call reset ('hwshd')
call surmat (rwdf,mesh,rdp2,mesh,rdp,0)
call endgr (3)
print*, ' Second subplot surface is complete'

```

c THIRD SUBPLOT

```

call reset ('xnonum')
call height (.150)
call physor (.5,.625)
call area2d (3.5,4.875)
call blsur
call volm3d (8.,8.,9.)
call x3name ('Frequency (Hz)',15)
call y3name ('',1)
call z3name ('Amplitude',9)
call vuangl (-90.,0.,30.)
call xintax
call zintax
call ynonum
call graf3d (0.,mfreq/4.,mfreq,0.,mtime/2.,mtime,zmin,
&(zmax-zmin)/3.,zmax)
print*, ' Third subplot axes are complete'
call reset ('hwshd')
call surmat (rwdf,mesh,rdp2,mesh,rdp,0)
call endgr (4)
print*, ' Third subplot surface is complete'

```

C FOURTH SUBPLOT

```

call reset ('ynonum')
call height (.125)
call physor (4.75,1.5)
call area2d (2.75,3.0)
call blsur
call xname ('Frequency (Hz)',15)
call yname ('Time (msec)',11)
call xintax
call yintax
call graf (0.,mfreq/4.,mfreq,0.,mtime/2.,mtime)
print*, ' Fourth subplot axes are complete'
call messag ('Contour at zmax/3',17,0.5,3.3)
call reset ('hwshd')
call conbgn
call zrange (zmax/3.,zmax/3.)
call conmak (rwdf,rdp2,rdp,1)
call conend
call conlin (1,'solid','nolabels',1,10)
call contur (1,'nolabels','draw')

```

```

call endgr (5)
print*, 'Fourth subplot surface is complete'
call endpl (0)
print*, 'Plotting complete'
print*
return
end

c
c      [rossano.thesis]plot3dsplit128.include
c
c      This subroutine uses disspla 11.0 to plot a series of
c      3 dimensional graphs of rwdf(256,128)
c

subroutine plot3dsplit128 (rwdf,outname,mtime,mfreq,dp,tcode,
&title,mnq,ampl)

real mtime,mfreq,zmax,zmin,rwdf(256,128),ampl
character*23,datetime,tcode,dpnum,amplif
character*50,title,outname,meani
integer*4 status
integer iplot_val,dp,mesh,rdp2,rdp,mnq

rdp=128
if (dp.eq.128) then
  dpnum='128 data points'
endif
if (dp.eq.256) then
  dpnum='256 data points'
endif
if (dp.eq.512) then
  dpnum='512 data points'
endif
if (dp.eq.1024) then
  dpnum='1024 data points'
endif
if (mnq.eq.1) then
  meani='Mean value removed'
endif
if (mnq.eq.2) then
  meani='Mean value not removed'
endif
amplif='Time amplified'
if (ampl.eq.0.0) amplif='Time amplified by 0.0'
if (ampl.eq.1.0) amplif='Time amplified by 1.0'
if (ampl.eq.2.0) amplif='Time amplified by 2.0'
if (ampl.eq.3.0) amplif='Time amplified by 3.0'
if (ampl.eq.4.0) amplif='Time amplified by 4.0'
if (ampl.eq.5.0) amplif='Time amplified by 5.0'
call rangel28 (rwdf,rdp,zmax,zmin)
print*, 'Enter limits for plotting (do not forget decimal)'
print*, '      ZMIN      ZMAX'
write(5,*) zmin,zmax
read(5,*) zmin,zmax
print*
mesh=1

```



```

rdp2=rdp*2
call reset ('all')
write(5,*) ' Do you want to view the plot on the screen or get
& a hardcopy?'
write(5,*) '                (1 for view, 2 for hardcopy)'
print*
read(5,*) iplot_val
print*
if (iplot_val .eq. 1) then
    print*, ' When finished viewing hit return key'
    call pgpx
endif
if (iplot_val .eq. 2) then
    print*, ' Please be patient'
    print*, ' This will take several minutes.'
    call LN03I
endif
call swissm
call hws hd
call chrpat (16)

```

c LABELS

```

call height (0.200)
call physor (0.5,0.625)
call area2d (7.5,9.75)
call alnmes (0.5,0.0)
call messag (title,100,7.5/2.,9.75)
call messag (outname,100,7.5/2.,9.5)
call reset ('alnmes')
call height (0.150)
call alnmes (0.0,1.0)
call messag (meani,50,5.0,0.0)
call messag (amplif,23,5.0,-0.2)
call messag (tcode,23,5.0,-0.4)
status = lib$date_time (datetime)
call messag (datetime,23,0.0,0.2)
call messag (dpnum,23,0.0,0.0)
call messag ('Reduced to 256 x 128',23,0.0,-0.2)
call messag ('Smoothed 10 x 10',23,0.0,-0.4)
call reset ('alnmes')
call endgr (1)
print*, ' Labels are complete'

```

c FIRST SUBPLOT

```

call height (.150)
call physor (.5,5.5)
call area2d (3.5,4.875)
call blsur
call volm3d (8.,8.,9.)
call x3name ('Frequency (Hz)',15)
call y3name ('Time (msec)',11)
call z3name ('Amplitude',9)
call vuangl (-50.,0.,30.)
call xintax
call yintax
call zintax

```

```

call graf3d (0.,mfreq/4.,mfreq,0.,mtime/2.,mtime,zmin,
&(zmax-zmin)/3.,zmax)
print*, ' First subplot axes are complete'
call reset ('hwshd')
call surmat (rwdf,mesh,rdp2,mesh,rdp,0)
call endgr (2)
print*, ' First subplot surface is complete'

```

c SECOND SUBPLOT

```

call height (.150)
call physor (4.25,5.5)
call area2d (3.5,4.5)
call blsur
call volm3d (8.,8.,9.)
call x3name ('',1)
call y3name ('Time (msec)',11)
call z3name ('Amplitude',9)
call vuangl (0.,0.,30.)
call yintax
call zintax
call xnonum
call graf3d (0.,mfreq/4.,mfreq,0.,mtime/2.,mtime,zmin,
&(zmax-zmin)/3.,zmax)
print*, ' Second subplot axes are complete'
call reset ('hwshd')
call surmat (rwdf,mesh,rdp2,mesh,rdp,0)
call endgr (3)
print*, ' Second subplot surface is complete'

```

c THIRD SUBPLOT

```

call reset ('xnonum')
call height (.150)
call physor (.5,.625)
call area2d (3.5,4.875)
call blsur
call volm3d (8.,8.,9.)
call x3name ('Frequency (Hz)',15)
call y3name ('',1)
call z3name ('Amplitude',9)
call vuangl (-90.,0.,30.)
call xintax
call zintax
call ynonum
call graf3d (0.,mfreq/4.,mfreq,0.,mtime/2.,mtime,zmin,
&(zmax-zmin)/3.,zmax)
print*, ' Third subplot axes are complete'
call reset ('hwshd')
call surmat (rwdf,mesh,rdp2,mesh,rdp,0)
call endgr (4)
print*, ' Third subplot surface is complete'

```

C FOURTH SUBPLOT

```

call reset ('ynonum')
call height (.125)
call physor (4.75,1.5)
call area2d (2.75,3.0)

```

```

call blsur
call xname ('Frequency (Hz)',15)
call yname ('Time (msec)',11)
call xintax
call yintax
call graf (0.,mfreq/4.,mfreq,0.,mtime/2.,mtime)
print*, 'Fourth subplot axes are complete'
call messag ('Contour at zmax/3',17,0.5,3.3)
call reset ('hwsd')
call conbgn
call zrange (zmax/3.,zmax/3.)
call conmak (rwdf,rdp2,rdp,1)
call conend
call conlin (1,'solid','nolabels',1,10)
call contour (1,'nolabels','draw')
call endgr (5)
print*, 'Fourth subplot surface is complete'
call endpl (0)
print*, 'Plotting complete'
print*
return
end

```

```

c
c      [rossano.thesis]plotcon32.include
c
c      This subroutine uses disspla 11.0 to plot contour
c      graphs of rwdf(64,32)
c

```

```

subroutine plotcon32 (rwdf,outname,mtime,mfreq,dp,tcode,
&title,mnq,ampl)

```

```

real mtime,mfreq,zmax,zmin,rwdf(64,32),ampi
character*23,datetime,tcode,dpnum,amplif
character*50,title,outname,meani
integer*4 status
integer iplot_val,dp,mesh,rdp2,rdp,mnq

```

```

rdp=32
if (dp.eq.128) then
  dpnum='128 data points'
endif
if (dp.eq.256) then
  dpnum='256 data points'
endif
if (dp.eq.512) then
  dpnum='512 data points'
endif
if (dp.eq.1024) then
  dpnum='1024 data points'
endif
if (mnq.eq.1) then
  meani='Mean value removed'
endif
if (mnq.eq.2) then
  meani='Mean value not removed'
endif

```

```

endif
amplif='Time amplified'
if (ampl.eq.0.0) amplif='Time amplified by 0.0'
if (ampl.eq.1.0) amplif='Time amplified by 1.0'
if (ampl.eq.2.0) amplif='Time amplified by 2.0'
if (ampl.eq.3.0) amplif='Time amplified by 3.0'
if (ampl.eq.4.0) amplif='Time amplified by 4.0'
if (ampl.eq.5.0) amplif='Time amplified by 5.0'
call range32 (rwdf,rdp,zmax,zmin)
print*, 'Enter limits for plotting (do not forget decimal)'
print*, '      ZMIN      ZMAX'
write(5,*) zmin,zmax
read(5,*) zmin,zmax
print*
mesh=1
rdp2=rdp*2
call reset ('all')
write(5,*) 'Do you want to view the plot on the screen or get
& a hardcopy?'
write(5,*) '      (1 for view, 2 for hardcopy)'
print*
read(5,*) iplot_val
print*
if (iplot_val .eq. 1) then
  print*, 'When finished viewing hit return key'
  call pgpx
endif
if (iplot_val .eq. 2) then
  print*, 'Please be patient'
  print*, 'This will take several minutes.'
  call LN03I
endif
call swissm
call hwshd
call chrpat (16)

```

c LABELS

```

call height (0.200)
call physor (0.5,0.625)
call area2d (7.5,9.75)
call alnmes (0.5,0.0)
call messag (title,100,7.5/2.,9.75)
call messag (outname,100,7.5/2.,9.5)
call messag ('Contours from zmax/6 to zmax',100,7.5/2.,9.25)
call reset ('alnmes')
call height (0.150)
call alnmes (0.0,1.0)
call messag (meani,50,5.0,0.0)
call messag (amplif,23,5.0,-0.2)
call messag (tcode,23,5.0,-0.4)
status = lib$date_time (datetime)
call messag (datetime,23,0.0,0.2)
call messag (dpnum,23,0.0,0.0)
call messag ('Reduced to 64 x 32',23,0.0,-0.2)
call messag ('Smoothed 10 x 10',23,0.0,-0.4)
call reset ('alnmes')

```

```

call endgr (1)
print*, ' Labels are complete'

c PLOT
call height (.125)
call physor (0.75,1.5)
call area2d (7.0,8.0)
call blsur
call xname ('Frequency (Hz)',15)
call yname ('Time (msec)',11)
call xintax
call yintax
call xticks (10)
call yticks (10)
call graf (0.,mfreq/4.,mfreq,0.,mtime/2.,mtime)
print*, ' Axes are complete'
call reset ('hwsht')
call conbgn
call zrange (zmax/6.,zmax)
call conmak (rwdf,rdp2,rdp,zmax/6.)
call conend
call conlin (1,'solid','nolabels',1,10)
call conlin (2,'dot','nolabels',1,10)
call conlin (3,'solid','nolabels',1,10)
call conlin (4,'dot','nolabels',1,10)
call conlin (5,'solid','nolabels',1,10)
call conlin (6,'dot','nolabels',1,10)
call contur (6,'nolabels','draw')
call endgr (2)
print*, ' Contours are complete'
call endpl (0)
print*, ' Plotting complete'
print*
return
end

c
c      [rossano.thesis]plotcon64. include
c
c      This subroutine uses disspla 11.0 to plot contour
c      graphs of rwdf(128,64)
c

subroutine plotcon64 (rwdf,outname,mtime,mfreq,dp,tcode,
&title,mnq,ampl)

real mtime,mfreq,zmax,zmin,rwdf(128,64),ampl
character*23,datetime,tcode,dpnum,amplif
character*50,title,outname,mean1
integer*4 status
integer iplot_val,dp,mesh,rdp2,rdp,mnq

rdp=64
if (dp.eq.128) then
  dpnum='128 data points'
endif
if (dp.eq.256) then

```

```

    dpnum='256 data points'
endif
if (dp.eq.512) then
    dpnum='512 data points'
endif
if (dp.eq.1024) then
    dpnum='1024 data points'
endif
if (mnq.eq.1) then
    meani='Mean value removed'
endif
if (mnq.eq.2) then
    meani='Mean value not removed'
endif
amplif='Time amplified'
if (ampl.eq.0.0) amplif='Time amplified by 0.0'
if (ampl.eq.1.0) amplif='Time amplified by 1.0'
if (ampl.eq.2.0) amplif='Time amplified by 2.0'
if (ampl.eq.3.0) amplif='Time amplified by 3.0'
if (ampl.eq.4.0) amplif='Time amplified by 4.0'
if (ampl.eq.5.0) amplif='Time amplified by 5.0'
call range64 (rwdf,rdp,zmax,zmin)
print*,' Enter limits for plotting (do not forget decimal)'
print*,'      ZMIN      ZMAX'
write(5,*) zmin,zmax
read(5,*) zmin,zmax
print*
mesh=1
rdp2=rdp*2
call reset ('all')
write(5,*) ' Do you want to view the plot on the screen or get
& a hardcopy?'
write(5,*) '      (1 for view, 2 for hardcopy)'
print*
read(5,*) iplot_val
print*
if (iplot_val .eq. 1) then
    print*,' When finished viewing hit return key'
    call pgpx
endif
if (iplot_val .eq. 2) then
    print*,' Please be patient'
    print*,' This will take several minutes.'
    call LN03I
endif
call swissm
call hws hd
call chrpat (16)

```

c LABELS

```

call height (0.200)
call physor (0.5,0.625)
call area2d (7.5,9.75)
call alnmes (0.5,0.0)
call messag (title,100,7.5/2.,9.75)
call messag (outname,100,7.5/2.,9.5)

```

```

call messag ('Contours from zmax/6 to zmax',100,7.5/2.,9.25)
call reset ('alnmes')
call height (0.150)
call alnmes (0.0,1.0)
call messag (meani,50,5.0,0.0)
call messag (amplif,23,5.0,-0.2)
call messag (tcode,23,5.0,-0.4)
status = lib$date_time (datetime)
call messag (datetime,23,0.0,0.2)
call messag (dpnum,23,0.0,0.0)
call messag ('Reduced to 128 x 64',23,0.0,-0.2)
call messag ('Smoothed 10 x 10',23,0.0,-0.4)
call reset ('alnmes')
call endgr (1)
print*, 'Labels are complete'

```

c PLOT

```

call height (.125)
call physor (0.75,1.5)
call area2d (7.0,8.0)
call blsur
call xname ('Frequency (Hz)',15)
call yname ('Time (msec)',11)
call xintax
call yintax
call xticks (10)
call yticks (10)
call graf (0.,mfreq/4.,mfreq,0.,mtime/2.,mtime)
print*, 'Axes are complete'
call reset ('hwshd')
call conbgn
call zrange (zmax/6.,zmax)
call conmak (rwdf,rdp2,rdp,zmax/6.)
call conend
call conlin (1,'solid','nolabels',1,10)
call conlin (2,'dot','nolabels',1,10)
call conlin (3,'solid','nolabels',1,10)
call conlin (4,'dot','nolabels',1,10)
call conlin (5,'solid','nolabels',1,10)
call conlin (6,'dot','nolabels',1,10)
call contur (6,'nolabels','draw')
call endgr (2)
print*, 'Contours are complete'
call endpl (0)
print*, 'Plotting complete'
print*
return
end

```

c

c [rossano.thesis]plotcon128.include

c

c This subroutine uses disspla 11.0 to plot contour
c graphs of rwdf(256,128)

c

```

subroutine plotcon128 (rwdf,outname,mtime,mfreq,dp,tcode,

```

```
&title,mnq,ampl)
```

```
real mtime,mfreq,zmax,zmin,rwdf(256,128),ampl
character*23,datetime,tcode,dpnum,amplif
character*50,title,outname,meani
integer*4 status
integer iplot_val,dp,mesh,rdp2,rdp,mnq
```

```
rdp=128
if (dp.eq.128) then
  dpnum='128 data points'
endif
if (dp.eq.256) then
  dpnum='256 data points'
endif
if (dp.eq.512) then
  dpnum='512 data points'
endif
if (dp.eq.1024) then
  dpnum='1024 data points'
endif
if (mnq.eq.1) then
  meani='Mean value removed'
endif
if (mnq.eq.2) then
  meani='Mean value not removed'
endif
amplif='Time amplified'
if (ampl.eq.0.0) amplif='Time amplified by 0.0'
if (ampl.eq.1.0) amplif='Time amplified by 1.0'
if (ampl.eq.2.0) amplif='Time amplified by 2.0'
if (ampl.eq.3.0) amplif='Time amplified by 3.0'
if (ampl.eq.4.0) amplif='Time amplified by 4.0'
if (ampl.eq.5.0) amplif='Time amplified by 5.0'
call rangel28 (rwdf,rdp,zmax,zmin)
print*,' Enter limits for plotting (do not forget decimal)'
print*,'      ZMIN      ZMAX'
write(5,*) zmin,zmax
read(5,*) zmin,zmax
print*
mesh=1
rdp2=rdp*2
call reset ('all')
write(5,*) ' Do you want to view the plot on the screen or get
& a hardcopy?'
write(5,*) '      (1 for view, 2 for hardcopy)'
print*
read(5,*) iplot_val
print*
if (iplot_val .eq. 1) then
  print*,' When finished viewing hit return key'
  call pgpx
endif
if (iplot_val .eq. 2) then
  print*,' Please be patient'
  print*,' This will take several minutes.'
```



```

        call LN03I
    endi
    call swissm
    call hwshd
    call chrpat (16)

```

c LABELS

```

    call height (0.200)
    call physor (0.5,0.625)
    call area2d (7.5,9.75)
    call alnmes (0.5,0.0)
    call messag (title,100,7.5/2.,9.75)
    call messag (outname,100,7.5/2.,9.5)
    call messag ('Contours from zmax/6 to zmax',100,7.5/2.,9.25)
    call reset ('alnmes')
    call height (0.150)
    call alnmes (0.0,1.0)
    call messag (meani,50,5.0,0.0)
    call messag (amplif,23,5.0,-0.2)
    call messag (tcode,23,5.0,-0.4)
    status = lib$date_time (datetime)
    call messag (datetime,23,0.0,0.2)
    call messag (dpnum,23,0.0,0.0)
    call messag ('Reduced to 256 x 128',23,0.0,-0.2)
    call messag ('Smoothed 10 x 10',23,0.0,-0.4)
    call reset ('alnmes')
    call endgr (1)
    print*, ' Labels are complete'

```

c PLOT

```

    call height (.125)
    call physor (0.75,1.5)
    call area2d (7.0,8.0)
    call blsur
    call xname ('Frequency (Hz)',15)
    call yname ('Time (msec)',11)
    call xintax
    call yintax
    call xticks (10)
    call yticks (10)
    call graf (0.,mfreq/4.,mfreq,0.,mtime/2.,mtime)
    print*, ' Axes are complete'
    call reset ('hwshd')
    call conbgn
    call zrange (zmax/6.,zmax)
    call conmak (rwdf,rdp2,rdp,zmax/6.)
    call conend
    call conlin (1,'solid','nolabels',1,10)
    call conlin (2,'dot','nolabels',1,10)
    call conlin (3,'solid','nolabels',1,10)
    call conlin (4,'dot','nolabels',1,10)
    call conlin (5,'solid','nolabels',1,10)
    call conlin (6,'dot','nolabels',1,10)
    call contur (6,'nolabels','draw')
    call endgr (2)
    print*, ' Contours are complete'

```

```

call endpl (0)
print*, 'Plotting complete'
print*
return
end

c
c      [rossano.thesis]pseudo.include
c
c      This subroutine smooths the WDF along both the time and
c      frequency axes
c

subroutine pseudo (dimt,dimf,rwdf,dp,dt,nn,mm)

integer dp,i,j,dimt,dimf,dp2,rdp,rdp2,nn,mm,nf,mt,nf2,mt2,
&ii,jj,L,LL,k,kk
real rwdf(256,128),hg(-25:25,-25:25),pi,dt,df,
&wdf(1100,550),coef,val
common /wdfc/ wdf

pi=4.*atan(1.)
df=1./(4.*dp*dt)
dp2=dp*2
rdp=dp/mm
rdp2=dp2/nn
print*, 'Smoothing 10 x 10'
nf=10
mt=10
nf2=nf*2
mt2=mt*2
val=1./((2.*pi*nf*df)*(2.*pi*mt*dt))
do 20 j=-mt2,mt2
  do 10 i=-nf2,nf2
    coef=-((j*j)/(2.*mt*mt))-((i*i)/(2.*nf*nf))
    hg(i,j)=val*exp(coef)
10  continue
20  continue
do 100 j=1,rdp
do 100 i=1,rdp2
100 rwdf(i,j)=0.0
do 500 i=1,dp2,nn
  ii=1+(i-1)/nn
  do 450 j=1,dp,mm
    jj=1+(j-1)/mm
    do 400 L=i-nf2,i+nf2
      LL=L
      if (L.lt.1) LL=1+dp2
      if (L.gt.dp2) LL=1-dp2
      do 350 k=j-mt2,j+mt2
        kk=k
        if (k.lt.1) kk=k+dp
        if (k.gt.dp) kk=k-dp
        rwdf(ii,jj)=rwdf(ii,jj)+wdf(LL,kk)*hg(L-i,k-j)
350  continue
400  continue
450  continue

```

```

500 continue
c This loop removes a plotting artifact
  do 600 j=1,rdp
    rwdf(rdp2-3,j)=0.0
    rwdf(rdp2-2,j)=0.0
    rwdf(rdp2-1,j)=0.0
    rwdf(rdp2,j)=0.0
600 continue
  return
end

c
c   [rossano.thesis]range32.include
c
c   This subroutine finds the maximum and minimum amplitudes
c   of array rwdf(64,32)
c

subroutine range32 (rwdf,dp,zmax,zmin)

real zmax,zmin,rwdf(64,32)
integer i,j,dp,dp2

dp2=dp*2
zmax=0.0
zmin=0.0
do 100 j=1,dp
  do 200 i=1,dp2
    if (rwdf(i,j).gt.zmax) zmax=rwdf(i,j)
    if (rwdf(i,j).lt.zmin) zmin=rwdf(i,j)
200 continue
100 continue
  return
end

c
c   [rossano.thesis]range64.include
c
c   This subroutine finds the maximum and minimum amplitudes
c   of array rwdf(128,64)
c

subroutine range64 (rwdf,dp,zmax,zmin)

real zmax,zmin,rwdf(128,64)
integer i,j,dp,dp2

dp2=dp*2
zmax=0.0
zmin=0.0
do 100 j=1,dp
  do 200 i=1,dp2
    if (rwdf(i,j).gt.zmax) zmax=rwdf(i,j)
    if (rwdf(i,j).lt.zmin) zmin=rwdf(i,j)
200 continue
100 continue
  return

```

```

end

c
c      [rossano.thesis] range128. include
c
c      This subroutine finds the maximum and minimum amplitudes
c      of array rwdf(256,128)
c

subroutine range128 (rwdf,dp,zmax,zmin)

real zmax,zmin,rwdf(256,128)
integer i,j,dp,dp2

dp2=dp*2
zmax=0.0
zmin=0.0
do 100 j=1,dp
  do 200 i=1,dp2
    if (rwdf(i,j).gt.zmax) zmax=rwdf(i,j)
    if (rwdf(i,j).lt.zmin) zmin=rwdf(i,j)
200  continue
100  continue
return
end

c
c      [rossano.thesis] reduce. include
c
c      This subroutine reduces the data in wdf to rwdf
c

subroutine reduce (dimt,dimf,dt,dp,nn,mm,rwdf,rval)

integer dimt,dimf,dp,dp2,i,j,ii,jj,nn,mm,rval
real wdf(1100,550),rwdf(256,128)
common /wdfc/ wdf

dp2=dp*2
df=1./(4.*dp*dt)

print*,' What do you want to reduce to?'
print*,'      1.  64 x 32'
print*,'      2. 128 x 64'
print*,'      3. 256 x 128'
read(5,*) rval

if (rval.eq.1) then
  nn=dp2/64
  mm=dp/32
endif
if (rval.eq.2) then
  nn=dp2/128
  mm=dp/64
endif
if (rval.eq.3) then
  nn=dp2/256
  mm=dp/128

```

```

endif
ii=0
jj=0
do 100 j=1,dp,mm
  jj=jj+1
  ii=0
  do 200 i=1,dp2,nn
    ii=ii+1
    rwdf(ii,jj)=wdf(i,j)
200  continue
100  continue
return
end

```

```

c
c      [rossano.thesis]size.include
c
c      This code is included in wigfun1.for and wigfun1b.for and their
c      subroutines. It provides an easy way to change the size of the
c      time plotting arrays.
c

```

```

c      real x(128),y(128),t(128)
c      real x(256),y(256),t(256)
c      real x(512),y(512),t(512)
c      real x(1024),y(1024),t(1024)
c      integer sizedp

```

```

      sizedp=512

```

```

c
c      [rossano.thesis]timesig.include
c
c      This subroutine modifies and plots the time signal
c

```

```

      subroutine timesig (dimt,ain,dp,dt,outname,tcode,
&title,s,sr,si,mnq,ampl)

```

```

      integer dp,mnq,wndwcode,anq,dimt,atvq
      real ain(dimt),sr(dimt),si(dimt),meanv,dt,zmax,ampl
      complex s(dimt)
      character*23 tcode
      character*50 title,titlehold,outname

```

```

      print*
      print*, ' Do you want to plot the raw time signal?'
      print*, '          (1 for yes, 2 for no)'
      read(5,*) atvq
      if (atvq.eq.1) then
        title='Raw Time Signal'
        call plot2d (ain,dt,dp,title,outname)
      endif
      print*
      call mean (dimt,ain,dp,meanv)
      print*, ' The mean value is'
      write(5,906) meanv
      print*

```

```

print*, ' Do you want to remove the mean from the time signal?'
print*, '          (1 for yes, 2 for no)'
read(*,*) mnq
if (mnq.eq.1) call meanr (dimt,ain,dp,meanv)
call maxamp (ain,dp,zmax)
print*
print*, ' The max amplitude is'
print*,zmax
print*
print*, ' You want this to be approx 1 in order to avoid
&plotting artifacts.'
ampl=1.0/zmax
print*
print*, ' Recommend an amplification of'
print*,ampl
print*
print*, ' Do you want to amplify the signal?'
print*, '          (1 for yes, 2 for no)'
read(5,*) atvq
if (atvq.eq.1) call amplify (dimt,ain,dp,ampl)
if (atvq.ne.1) ampl=0.0
print*
print*, ' Do you want to window the time signal?'
print*, '          (1 for yes, 2 for no)'
read(5,*) atvq
if (atvq.eq.1) then
    call window (dimt,ain,dp,dt,wndwcode)
    if (wndwcode.eq.2) tcode='Hanning window time'
    if (wndwcode.eq.3) tcode='Hamming window time'
endif
print*
print*, ' Do you want to plot the modified time signal?'
print*, '          (1 for yes, 2 for no)'
read(5,*) atvq
if (atvq.eq.1) then
    title='Modified Time Signal'
    call plot2d (ain,dt,dp,title,outname)
endif
title='Wigner Distribution'
print*
print*, ' Do you want to make the time signal analytic?'
print*, '          (1 for yes, 2 for no)'
read(5,*) anq
if (anq.eq.1) then
    title='Wigner-Ville Distribution'
    call analytic (dimt,ain,s,dp)
    do 100 j=1,dp
        sr(j)=ain(j)
        si(j)=aimag(s(j))
100    continue
    print*
    print*, ' Do you want to plot the analytic time signal?'
    print*, '          (1 for yes, 2 for no)'
    read(5,*) atvq
    if (atvq.eq.1) then
        print*

```

```

        if (dp.ge.512) then
            print*, ' The STD00001.dat file may be too large to print.'
            print*, ' For a hardcopy run [rossano.thesis]wigfunlb.for'
            print*
            print*, ' Do you want to continue with analytic plotting
& here?'
            print*, '                (1 for yes, 2 for no)'
            read(5,*) atvq
            if (atvq.eq.2) go to 300
        endif
        titlehold=title
        title='Analytic Time Signal'
        call plot2d2 (sr,si,dt,dp,title,outname,mnq)
        title=titlehold
    endif
endif
300  if (anq.eq.2) then
        do 200 j=1,dp
            sr(j)=ain(j)
            si(j)=0.0
            s(j)=cmplx(sr(j),si(j))
200    continue
        endif
        return
906  format(f16.8)
        end
c
c      [rossano.thesis]wigh.include
c
c      This subroutine calculates the WDF
c

subroutine wigh (dimt,dimf,s,c,df,dt,dp2,dp)

integer dimt,dimf,i,j,dp2,dp
complex s(dimt),c(dimf)
real wdf(1100,550),df,dt
common /wdfc/ wdf

do 100 j=1,dp
    call corr (dimt,dimf,s,j,dt,c,dp)
    call fft (dimf,c,dp2,0)
    do 200 i=1,dp2
        wdf(i,j)=real(c(i))
200    continue
100 continue
    return
end

c
c      [rossano.thesis>window.include
c
c      This subroutine calls the available windowing functions
c

subroutine window (dimt,ain,dp,dt,wndwcode)

```

```

integer atvq,dp,wndwcode,dimt
real ain(dimt),dt

print*
print*, ' Which window would you like to apply?'
print*, '      1. None'
print*, '      2. Hanning'
print*, '      3. Hamming'
read(5,*) atvq
if (atvq.eq.1) go to 100
if (atvq.eq.2) then
    call hanning (dimt,ain,dp,dt)
    wndwcode=2
endif
if (atvq.eq.3) then
    call hamming (dimt,ain,dp,dt)
    wndwcode=3
endif
100 return
end

```

D. DATA CONVERSION PROGRAM

```

c
c      [rossano.thesis]convert.for
c
c      This program converts the experimental input data file obtained
c      from HP Vista into the format used for WIGFUN1.FOR
c

```

```

real tim(2),amp(2)
integer modq,i,n
character*20 inname,outname
character*1 c(6)

print*
print*, ' This program converts HP Vista data input format into'
print*, ' the data format used in
& [rossano.thesis]WIGFUN1.FOR'
print*
print*, ' Enter HP Vista input filename'
read(5,901) inname
print*
print*, ' Has the last line been modified so that 9.999999'
print*, ' is in each column to indicate an end of file?'
print*, ' An example follows:'
print*, ' ( 4.960938e-01, 7.668274e-01)
& ( 4.980469e-01, 1.742336e-01)'
print*, ' ( 5.000000e-01, -5.128824e-01)
& ( 9.999999 , 9.999999 )'
print*, ' ( 9.999999 , 9.999999 )
& ( 9.999999 , 9.999999 )'
print*, ' (1 for yes, 2 for no)'
read(5,902)modq
if (modq.eq.2) then
    print*, ' You need to edit the HP Vista file'
    print*, ' Good bye'

```



```

        go to 900
    endif
    print*
    print*, ' Enter output filename (i.e. sin62.5e)'
    read(5,901) outname
    open (unit=4,file=inname,status='old')
    rewind 4
    open (unit=7,file=outname,status='new')
    print*
    print*, ' Delta t ='
    print*
    n=0
    do 100 i=1,10000
        read(4,903) c(1),tim(1),c(2),amp(1),c(3),c(4),tim(2),
&          c(5),amp(2),c(6)
        if (tim(1).eq.9.999999) then
            if (amp(1).eq.9.999999) go to 200
        endif
        n=n+1
        write(7,904) tim(1),amp(1)
        if (tim(2).eq.9.999999) then
            if (amp(2).eq.9.999999) go to 200
        endif
        n=n+1
        write(7,904) tim(2),amp(2)
        if (n.eq.2) go to 300
        if (n.eq.102) go to 300
        if (n.eq.202) go to 300
        if (n.eq.302) go to 300
        if (n.eq.402) go to 300
        if (n.eq.502) go to 300
        if (n.eq.602) go to 300
        if (n.eq.702) go to 300
        if (n.eq.802) go to 300
        if (n.eq.902) go to 300
100    continue
        print*
        print*, ' There are more points remaining in HP Vista data file'
        print*, ' increase the loop size'
200    write(7,904) 9999.,9999.
        print*
        print*, ' Total number of data points is'
        write(5,905) n
        go to 900
300    dt=tim(2)-tim(1)
        print*,dt
        go to 100
900    close (unit=4)
        close (unit=7)
901    format (a20)
902    format (i1)
903    format (3x,a1,e15.6,a1,e15.6,a1,2x,a1,e15.6,a1,e15.6,a1)
904    format (2x,e16.8,5x,e16.8)
905    format (1x,i10)
    end

```

APPENDIX B. DATA TRANSFER FROM SOURCE TO VAX COMPUTER

A. DATA SOURCE TO HP3565 COMPUTER

Numerous sources of data were used. Analog sources included function generators and FM tape recordings of accelerometer, velocity, and pressure transducers. Another source of data resulted from direct measurements of vibrations of machinery in the laboratory. All signals were fed into an input module of an HP3565 computer based on the HP9000 350 CPU which included a dynamic signal analyzer. Hewlett-Packard's HP-Vista software was used to control the hardware. The input module was calibrated and ranged for the applicable voltages of the input signal. The digital filters were set to allow the frequencies of interest to pass. The primary purpose of the HP3565 computer system was to digitize and filter the data.

B. HP3565 COMPUTER TO PC

Once the data was digitized, a copy of the raw data was printed out for use in verifying the data upon arrival in the VAX computer. The digitized data was transferred from the HP3565 to a personal computer (PC) as a printed ASCII file output. The actual data transfer took place over an RS232 cable connecting the laser printer port on the HP3565 to the PC communications port.

C. PC TO VAX

At the PC the data file was transferred to a 5.25" floppy disc. This data was downloaded to the VAX computer in the Mechanical Engineering CAD/CAE Lab via the terminal available there.

D. EDITING DATA FILE UPON ARRIVAL IN THE VAX

The data file which arrived in the VAX included a banner page and file header information which had to be manually removed by editing. In addition, there were form-feed characters throughout the file which had to be removed. The data in the VAX was compared with the data printouts made by the HP3565. On the average there were 3 or 4 corrections to a 1024 data point file due to the high data transfer rate across the RS232 cable (9600 Baud). A slower data transfer rate could have been used to improve the accuracy, but the transfer time would have taken much longer. A final line of characters was added to the data file to serve as an end of file indication. A computer

program, CONVERT.FOR (the last program in Appendix A), was used to convert the data from the edited HP3565 format into the format used in subroutine DATAIN.INCLUDE. The data format conversion program also provided a final check to ensure that all data points had been included by counting the number of data points. A block size setting of 1024 real or 8192 real on the HP3565 resulted in 1024 or 8192 data points, respectively.

LIST OF REFERENCES

1. Naval Postgraduate School Report NPS 69-90-03, *Submarine-Installed Machinery Monitoring and Diagnostics: A State-of-the-Art Review*, by J. Robinson, G. Rossano, and Y. Shin, Mar 1990.
2. Hewlett-Packard, *Dynamic Signal Analyzer Applications: Effective Machinery Maintenance Using Vibration Analysis*, HP Application Note 243-1.
3. Bastiaans, M.J., "The Wigner Distribution Function Applied to Optical Signals and Systems," *Optics Communications*, vol. 25, no. 1, pp. 26-30, Apr 1978.
4. Bastiaans, M.J., "Wigner Distribution Function and its Application to First-Order Optics," *Journal of the Optical Society of America*, vol. 69, no. 12, pp. 1710-1716, Dec 1979.
5. Bartelt, H.O., Brenner, K.H. and Lohmann, A.W., "The Wigner Distribution Function and its Optical Production," *Optics Communications*, vol. 32, no. 1, pp. 32-38, Jan 1980.
6. Riley, M., *Speech Time-Frequency Representations*, Kluwer Academic Publishers, 1989.
7. Velez, E. and Absher, R., "Transient Analysis of Speech Signals using the Wigner Time-Frequency Representation," IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 4, pp. 2242-2245, May 1989.
8. Wahl, T., and Bolton, J., "The Use of the Wigner Distribution to Analyze Structural Impulse Responses," International Congress on Recent Developments in Air and Structure-Borne Sound, Mar 1990.
9. Flandrin, P., Garreau, D., and Puyal, C., "Improving Monitoring of PWR Electrical Power Plants 'In Core' Instrumentation with Time- Frequency Signal Analysis,"

IEEE International Conference on Acoustics, Speech, and Signal Processing. vol. 4, pp. 2246-2249, May 1989.

10. Forrester, B., "Analysis of Gear Vibration in the Time-Frequency Domain," Proceedings of the 44th Meeting of the Mechanical Failures Prevention Group, pp. 225-234, Apr 1990.
11. Wigner, E., "On the Quantum Correction for Thermodynamic Equilibrium," *Physical Review*, vol. 40, pp. 749-759, Jun 1932.
12. Claasen, T. and Mecklenbrauker, W., "The Wigner Distribution - A Tool for Time-Frequency Signal Analysis Part I: Continuous-Time Signals," *Philips Journal of Research*, vol. 35, no. 3, pp. 217-250, 1980.
13. Claasen, T. and Mecklenbrauker, W., "The Wigner Distribution - A Tool for Time-Frequency Signal Analysis Part II: Discrete-Time Signals," *Philips Journal of Research*, vol. 35, nos. 4 5, pp. 276-300, 1980.
14. Claasen, T. and Mecklenbrauker, W., "The Wigner Distribution - A Tool for Time-Frequency Signal Analysis Part III: Relations with other Time-Frequency Signal Transformations," *Philips Journal of Research*, vol. 35, no. 6, pp. 372-389, 1980.
15. Yen, N., "Time and Frequency Representation of Acoustic Signals by Means of the Wigner Distribution Function: Implementation and Interpretation," *Journal of the Acoustical Society of America*, vol. 81, no. 6, pp. 1841-1850, Jun 1987.
16. Ville, J., "Theorie et Applications de la Notion de Signal Analytique," *Cables et Transmission*, vol. 2a, no. 1, pp. 61-74, 1948.
17. Boashash, B. and Black, P.J., "An Efficient Real-Time Implementation of the Wigner-Ville Distribution," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-35, no. 11, pp. 1611-1618, Nov 1987.

18. Boashash, B., "Note on the Use of the Wigner Distribution for Time- Frequency Signal Analysis," IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 36, no. 9, pp. 1518-1521, Sep 1988.
19. Jones, D. and Parks, T., "A Resolution Comparison of Several Time- Frequency Representations," IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 4, pp. 2222-2225, May 1989.
20. Andrieux, J.C. et al., "Optimum Smoothing of the Wigner-Ville Distribution," IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. ASSP-35, no. 6, pp. 764-768, Jun 1987.
21. Kadambe, S., Boudreaux-Bartels, G.F., and Duvaut, P., "Window Length Selection for Smoothing the Wigner Distribution by Applying an Adaptive Filter Technique," IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 4, pp. 2226-2229, May 1989.
22. Sun, M., et. al., "Elimination of Cross-Components of the Discrete Pseudo Wigner Distribution via Image Processing," IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 4, pp. 2230-2233, May 1989.
23. Naval Underwater Systems Center Technical Report 8225, *Wigner Distribution Function: Relation to Short-Term Spectral Estimation, Smoothing, and Performance in Noise*, by A. Nuttall, Feb 1988.
24. Computer Associates International, Inc., *CA - DISSPLA[®]*, version 11.0, 1988.
25. Strum, R., and Kirk, D., *First Principles of Discrete Systems and Digital Signal Processing*, p. 560, Addison-Wesley Publishing Co., 1988.
26. Bendat, J., and Piersol, A., *Engineering Applications of Correlation and Spectral Analysis*, John Wiley & Sons, p. 74, 1980.

BIBLIOGRAPHY

Bouachache, B. and Rodriguez, F., "Recognition of Time-Varying Signals in the Time-Frequency Domain by Means of the Wigner Distribution," IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 2, pp. 22.5.1-22.5.4, Mar 1984.

Boudreaux-Bartels, G.F. and Parks, T.W., "Signal Estimation Using Modified Wigner Distributions," IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 2, pp. 22.3.1- 22.3.4, Mar 1984.

Boudreaux-Bartels, G.F. and Parks, T.W., "Time-Varying Filtering and Signal Estimation Using Wigner Distribution Synthesis Techniques," IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. ASSP-34, no. 3, pp. 442-451, Jun 1986.

Chan, D., "A Non-Aliased Discrete-Time Wigner Distribution for Time-Frequency Signal Analysis," IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 2, pp. 1333-1336, May 1982.

Claasen, T. and Mecklenbrauker, W., "The Aliasing Problem in Discrete-Time Wigner Distributions," IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. ASSP-31, no. 5, pp. 1067-1072, Oct 1983.

Cohen, F.S., Boudreaux-Bartels, G.F., and Kadambe, S., "Tracking of Unknown Non-Stationary Chirp Signals using Unsupervised Clustering in the Wigner Distribution Space," IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 4, pp. 2180-2183, Apr 1988.

Cohen, L., "On a Fundamental Property of the Wigner Distribution," IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. ASSP-35, no. 4, pp. 559-561, Apr 1987.

Cohen, L., "Wigner Distribution for Finite Duration or Band-Limited Signals and Limiting Cases," IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. ASSP-35, no. 6, pp. 796-806, Jun 1987.

Flandrin, P., "Maximum Signal Energy Concentration in a Time-Frequency Domain," IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 4, pp. 2176-2179, Apr 1988.

Griffin, C., Rao, P., and Taylor, F., "Roundoff Error Analysis of the Discrete Wigner Distribution using Fixed-Point Arithmetic," IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 2, pp. 869-871, May 1989.

Hewlett-Packard, *The Fundamentals of Signal Analysis*, HP Application Note 243.

Hippenstiel, R. and Oliveira, P., "Contributions to Time Varying Spectrum Estimation using the Instantaneous Power Spectrum (IPS)," IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 4, pp. 2093-2096, May 1989.

Janssen, A., "On the Locus and Spread of Pseudo-Density Functions in the Time-Frequency Plane," *Philips Journal of Research*, vol 37, no. 3, pp. 79-110, 1982.

Janssen, A., "Wigner Weight Functions and Weyl Symbols of Non-Negative Definite Linear Operators," *Philips Journal of Research*, vol. 44, no. 1, pp. 7-42, 1989.

Lee, C. and Cohen, L., "Instantaneous Mean Quantities in Time-Frequency Analysis," IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 4, pp. 2188-2191, Apr 1988.

Mark, W., "Power Spectrum Representation for Nonstationary Random Vibration," *Random Vibration - Status and Recent Developments*, Elsevier, pp. 211-240, 1986.

Naval Underwater Systems Center Technical Report 8317, *The Wigner Distribution Function with Minimum Spread*, by A. Nuttall, Jun 1988.

Oliveira, P., *Instantaneous Power Spectrum*, Master's Thesis, Naval Postgraduate School, Monterey, CA, Mar 1989.

Robinson, J., *Machinery Vibration Analysis Using Statistical Parameters of the Time Domain Signal*, Master's Thesis, Naval Postgraduate School, Monterey, CA, Dec 1989.

Romberg, T., Cassar, A., and Harris, R., "A Comparison of Traditional Fourier and Maximum Entropy Spectral Methods for Vibration Analysis," *Transactions of the ASME Journal of Vibration, Acoustics, Stress, and Reliability in Design*, vol. 106, no. 1, pp. 36-39, Jan 1984.

Saleh, B. and Subotic, N., "Time-Variant Filtering of Signals in the Mixed Time-Frequency Domain," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-33, no. 6, pp. 1479-1485, Dec 1985.

Stamm, J., *Machinery Diagnostics via Mechanical Vibration Analysis using Spectral Analysis Techniques*, Master's Thesis, Naval Postgraduate School, Monterey, CA, Sep 1988.

Yu, K. and Cheng, S., "Signal Synthesis from Pseudo-Wigner Distribution and Applications," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-35, no. 9, pp. 1289-1302, Sep 1987.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Dean of Science and Engineering, Code 06 Naval Postgraduate School Monterey, CA 93943-5004	2
4.	Research Administrations Office, Code 012 Naval Postgraduate School Monterey, CA 93943-5004	1
5.	Department Chairman, Code 69 Department of Mechanical Engineering Naval Postgraduate School Monterey, CA 93943-5004	1
6.	Naval Engineering Curricular Office, Code 34 Department of Mechanical Engineering Naval Postgraduate School Monterey, CA 93943-5004	1
7.	Professor Y. S. Shin, Code 69Sg Department of Mechanical Engineering Naval Postgraduate School Monterey, CA 93943-5004	3
8.	Professor J. F. Hamilton, Code 69Ha Department of Mechanical Engineering Naval Postgraduate School Monterey, CA 93943-5004	1
9.	CAPT D. P. Mahoney, USN Naval Sea Systems Command Sea 56 Washington, DC 20362	1
10.	Mr. G. Reid Naval Surface Warfare Center Silver Spring, MD 20903-5000	1

11. LT G. W. Rossano, USN 1
63 Hundreds Circle
Wellesley Hills, MA 02181
12. Professor R. D. Hippenstiel, Code 62Hi 1
Department of Electrical and Computer Engineering
Naval Postgraduate School
Monterey, CA 93943-5004
13. Mrs. R. Holtzman 3
Naval Sea Systems Command, PMS 390TC25
Washington, DC 20362-5101
14. Ms. D. Cuomo 1
Naval Sea Systems Command, PMS390TC26
Washington, DC 20362-5101
15. Mr. A. Pride 1
Submarine Maintenance and Support Office
Naval Sea Systems Command, PMS390
Washington, DC 20362-5101
16. Mr. D. Bills 1
Naval Sea Systems Command, 56W13
Washington, DC 20362-5101
17. Dr. K. Ng 1
Naval Underwater Systems Command
Newport, RI 02841
18. Mr. W. McInnis, Code 8321 1
Naval Underwater Systems Command
Newport, RI 02841
19. LCDR R. J. Martin, USN 1
Mechanical and Electrical Submarine Technology Program
1515 Wilson Blvd., Suite 705
Arlington, VA 22209
20. Mr. B. Marshall 1
Naval Ship System Engineering Station
Philadelphia, PA 19112-5083
21. Mr. K. R. Rossano 1
63 Hundreds Circle
Wellesley Hills, MA 02181
22. LT J. D. Robinson, USN 1
2505 Rolling Meadows Drive
Gautier, MS 39553-4929